Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Επεξεργασιας Φυσικης Γλωσσας

Online Learning for Automatic Quality Estimation of Machine Translation Output

Διπλωματικη Εργασια

Αντώνιος Γ. Αναστασοπουλος

Επιβλέπων:
Γιάνης Μαϊστρος
Επίκουρος Καθηγητής ΕΜΠ

Εξωτερικοί Επιβλέποντες:
Matteo Negri
Marco Turchi
(Fondazione Bruno Kessler)

Αθήνα, Απρίλιος 2014

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Επεξεργασιας Φυσικης Γλωσσας

Online Learning for Automatic Quality Estimation of Machine Translation Output

Διπλωματικη Εργασια

Αντώνιος Αναστασοπουλος

Επιβλέπων:
Γιάνης Μαΐστρος
Επίκουρος Καθηγητής ΕΜΠ

Εξωτερικοί Επιβλέποντες:
Matteo Negri
Marco Turchi
(Fondazione Bruno Kessler)

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 9η Απριλίου 2014.

Γιάνης Μαΐστρος	Παγουρτζής Αριστείδης	Στάμου Γιώργος
Επίκ. Καθηγητής ΕΜΠ	Επίκ. Καθηγητής ΕΜΠ	Επίκ. Καθηγητής ΕΜΠ

Αναστασόπουλος Αντώνιος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright ©Anastasopoulos Antonios, 2013.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξολοκλήρου ή μέρους αυτής, για εμπορικό ή κερδοσκοπικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για εμπορικό-κερδοσκοπικό σκοπό πρέπει να απευθύνονται αποκλειστικά στους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτή την εργασία εκφράζουν τους συγγραφείς και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου συμπεριλαμβανόμενων Σχολών, Τομέων και Μονάδων αυτού.

Abstract

Online Learning for Automatic Quality Estimation of Machine Translation Output

by Antonios Anastasopoulos

The automatic estimation of Machine Translation output quality is a hard task, where the selection of the appropriate algorithm and the most predictive features often plays a crucial role. When moving from controlled lab evaluations to real-life scenarios the task becomes even harder. For current Machine Translation Quality Estimation systems, additional complexity comes from the difficulty to model user and domain changes. Systems' instability with respect to data coming from different distributions, in fact, calls for adaptive solutions that quickly react to new operating conditions. To tackle this issue we propose an online framework for adaptive Quality Estimation, targeting reactivity and robustness to user and domain changes.

We experiment with different online machine learning techniques like Online Support Vector Regression, Passive Aggressive Algorithms and Online Gaussian Processes. We also perform contrastive experiments with two language pairs, English-Spanish and English-Italian, in different testing conditions. The outcome of the experiments demonstrates the effectiveness of this approach.

Keywords: *Quality Estimation, Machine Translation, Online (Adaptive) Learning*

"Translation is like rewriting from scratch. In has to be extremely sensitive to not just the syntax but the deeper linguistic features. It involves translating from one culture to another, so that it resonates with someone of that other culture."

Gao Xingjian
Literature Nobel Prize 2013

Acknowledgements

Η διπλωματική αυτή εργασία εκπονήθηκε τόσο στο Human Language Technologies group στο Fondazione Bruno Kessler (FBK) στο Τρέντο (Ιταλία), στα πλαίσια του ερευνητικού έργου MateCat (§1.1.1), καθώς και στα πλαίσια των ερευνητικών δραστηριοτήτων του Εργαστηρίου Επεξεργασίας Φυσικής Γλώσσας του Εθνικού Μετσόβιου Πολυτεχνείου.

Αρχικά, θα ήθελα να ευχαριστήσω τον Καθηγητή Γιάνη Μαΐστρο για τις συμβουλές του, τη διαρκή υποστήριξή του, και για την ευκαιρία που μου έδωσε να ασχοληθώ ερευνητικά στον τομέα της Επεξεργασίας Φυσικής Γλώσσας. Επιπλέον, θα ήθελα να ευχαριστήσω θερμά τους Matteo Negri και Marco Turchi, καθώς και τα υπόλοιπα μέλη του HTL group, για την εμπιστοσύνη τους στην ανάθεση του ερευνητικού αυτού έργου, τις ιδέες και τις γνώσεις τους που συνέβαλαν στην πραγματοποίηση της πρακτικής άσκησης, καθώς και τη συγγραφή και εποπτεία της διπλωματικής αυτής εργασίας.

Επίσης, θερμές ευχαριστίες θα ήθελα να απευθύνω στον José G.C. de Souza, για την καθοριστική συμβολή του στη δημιουργία του λογισμικού καθώς και την πραγματοποίηση των πειραμάτων. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την εμψύχωση που μου παρείχαν καθ΄ όλη τη διάρκεια των σπουδών μου.

This thesis is the result of work conducted in both the Human Language Technologies group at Fondazione Bruno Kessler (FBK), Trento, as part of the MateCat project and the research activity of the Natural Language Processing Laboratory of the National Technical University of Athens.

I would like to thank professor Yanis Maistros for his advice, his constant support and for guiding me into the world of Natural Language Processing. In addition, I would like to thank Matteo Negri and Marco Turchi, as well as the rest of the HLT group at FBK, for trusting me with this project, helping with their ideas and expertise during my internship and afterwards, as well as for supervising this thesis.

Furthermore, special thanks and credit should be given to José G.C. de Souza, without whom the development of the software and the experiments would have been much harder. Finally, I would like to thank my family and friends for their great support throughout my studies.

Αναστασόπουλος Αντώνιος

Αθήνα, Απρίλιος 2014

Contents

List of Figures

List of Tables

Abbreviations

MT	**M**achine **T**ranslation
QE	**Q**uality **E**stimation
CAT	**C**omputer **A**ssisted **T**ranslation
TM	**T**ranslation **M**emory
SMT	**S**tatistical **M**achine **T**ranslation
MAE	**M**ean **A**bsolute **E**rror
RMSE	**R**oot **M**ean **S**quare **E**rror
SVM	**S**upport **V**ector **M**achine
SVR	**S**upport **V**ector **R**egression
PA	**P**assive **A**ggressive
GP	**G**aussian **P**rocess

Chapter 1

Introduction

1.1 Motivation

Recent advances in computer science and, more precisely, statistical natural language processing have enabled the creation of even more reliable Statistical Machine Translation (MT) tools, which produce results good enough to be used in a professional translation workflow.

As the SMT systems evolve, one of the next steps is to try to automatically evaluate the translated data, instead of relying on costly human annotations. This line of research has formed the Quality Estimation task, further described in §2.5.

The first goal of this project was to implement a system that will perform online Quality Estimation, suitable for integration with the core system of the MateCat project. The second goal was to test the performance of the system, in order to verify that indeed such a QE system would be beneficiary for the CAT-tool that is developed by the Matecat project.

1.1.1 The MateCat project

MateCat is a project trying to effectively and ergonomically integrate Machine Translation within the human translation workflow.

While today MT is mainly trained with the objective of creating the most comprehensible output, MateCat targets MT technology that will minimize the translator's post-editing effort.

To this end, MateCat is developing an enhanced web-based CAT tool that will offer new MT capabilities, such as automatic adaptation to the translated content, online learning from user corrections, and automatic quality estimation.

The project builds on state-of-the-art MT and CAT technologies created by the project members (Fondazione Bruno Kessler, Translated, Université du Maine, University of Edinburgh). Such technologies include Moses, the most popular open source statistical MT toolkit, and MyMemory, the world's largest Translation Memory (TM) built collaboratively via MT and human contributions.

In order to optimally integrate MT into the CAT workflow and enhance translator's productivity and user experience, MateCat attempts to innovate by creating new operating conditions for MT, so as to match the CAT application, such as:

- Self-tuning MT, that could be domain- or project-adaptive and perform document analysis in order to improve translation coherence.
- User-adaptive MT, that could adapt to user feedback in an on-line and realtime fashion, but would also be context-aware, augmented with lexical/syntactic constraints.
- Informative MT, that would learn and help with terminology, provide confidence measures and would also produce enriched MT output, by displaying alternative outputs or highlighting possible parts that need to be edited.

The ultimate goal is to create new CAT technology that will significantly enhance the productivity and user experience of professional translators.

1.2 Organization

This booklet is organized as follows:

- Chapter 2 presents an overview of the theoretical background needed. The basic concepts of Machine Translation and Quality Estimation are presented.
- Chapter 3 describes the theoretical and algorithmic aspects of the machine learning techniques that we used.
- Chapter 4 contains the system specifications and requirements, describing the libraries and the core system that we used for the Online Quality Estimation component.
- Chapters 5 and 6 present the experiments that test the performance of our system, attempting, in the end, to provide a suggestion for the best configuration of the system.
- Chapter 7, finally, summarizes the main results and findings of this work.

Chapter 2

Quality Estimation for Statistical Machine Translation in a Computer-assisted Framework

2.1 Introduction

This chapter provides a short overview of the development of the Machine Translation field, with an emphasis on its integration in the Computer-Assisted Translation framework.

In addition, the task of Quality Estimation is defined, presenting the various techniques that have been developed to estimate the MT output quality. The most recent developments on the task, as part of recent evaluation campaigns, are presented.

Finally, the last section describes the challenges that haven't been addressed so far, and with which this thesis tries to cope.

2.2 Machine Translation Overview

Machine Translation (MT) is the field investigating the use of automated methods (*software*) in order to translate text or speech from one language to another. It was one of the first applications envisioned for computers, starting with Warren Weaver in 1949 with his *"Translation Memorandum"* and other articles (Weaver, 1955), even before people had any idea of what computers might be capable of.

Following the initial efforts by IBM researchers (Brown et al., 1990), several approaches have been developed in order to deal with the challenges of MT. Such approaches were:

- Word-for-word translation, which uses bilingual dictionaries to translate each word of a text. However, this approach results in low quality translations, since problems like different word order or word sense ambiguity emerge.

- Rule-Based approaches were the first approaches and the most dominant paradigms until the 1990s. A simple overview of an RB-system would be:

$$source \xrightarrow{transform} intermediate\ representation \xrightarrow{transform} target$$

  with the transformation being decided by representation rules. These are also called *Syntactic transfer* approaches, since they construct transfer rules for syntactic trees across languages, dealing with the word order problem. However, this is not only difficult to implement for all possible pairs of languages, due to the need of various resources, but there are always cases of syntactic mismatches between the languages.

- Interlingual approaches. The "interlingua" is an intermediate representation between the languages, like in rule-based approaches. However, the *interlingual* approaches try to use a logical form which represents the semantics of the sentence as an intermediate step between the languages. Defining this *interlingua*, though, is by itself a very difficult problem.

- Statistical translation. This approach uses statistical methods in order to find the most probable translation, given the sentence to be translated, using information derived from parallel corpora. It is currently the most commonly used approach and is discussed in the next section.

2.3 Statistical Machine Translation (SMT)

2.3.1 Overview

Given a foreign language $\mathcal{F}$ and a sentence f, the SMT approach tries to find the most probable sentence $\hat{s}$ in the translation target language $\mathcal{S}$, out of all possible translations s. This means finding the sentence $\hat{s}$ with the highest probability $p(s|f)$ where:

$$\hat{s} = arg\ max_s\ p(s|f)$$

Using the following equation from the Bayes rule:

$$p(s|f) = \frac{p(s)p(f|s)}{p(f)}$$

the previous is transformed to

$$\hat{s} = arg\ max_s\ p(s)p(f|s)$$

This can be broken down into two components:

- $p(s)$ is the *"Language Model"*, which assigns higher probability to fluent/grammatical sentences and is computed based on monolingual corpora of the target language $\mathcal{S}$, using n-grams probabilities.
- $p(f|s)$ is the *"Translation Model"* which assigns higher probability to sentences with corresponding meaning and is estimated based on parallel corpora of the two languages $\mathcal{F}$ and $\mathcal{S}$, commonly using alignment probabilities, phrase tables or more advanced methods.

2.3.2 Comparison to other approaches

Some of the advantages of the statistical approach for performing MT are the following:

- it is data driven, and therefore it does not need human annotators, linguists, etc.
- it is language independent, since it can be created for practically any language pair that has enough training data.
- its building blocks are human translated blocks, resulting in state-of-the-art translations when large data sets are available.

As disadvantages of SMT could be considered the facts that:

- it doesn't take advantage of all available information, since eg. it does not explicitly deal with syntax,
- it needs large data sets for high quality performance, which might not be available for all language pairs.

It is worth noting that recent advanced models try to deal with the first disadvantage, by also taking a *"syntax model"* into account (Charniak et al., 2003, Yamada and Knight, 2001). More recent approaches might also use N-grams with syntactic information (Crego and Marino, 2007), shallow syntactic information as input (Crego and Habash, 2008) or create rules to translate the paths of dependency trees (Lin, 2004).

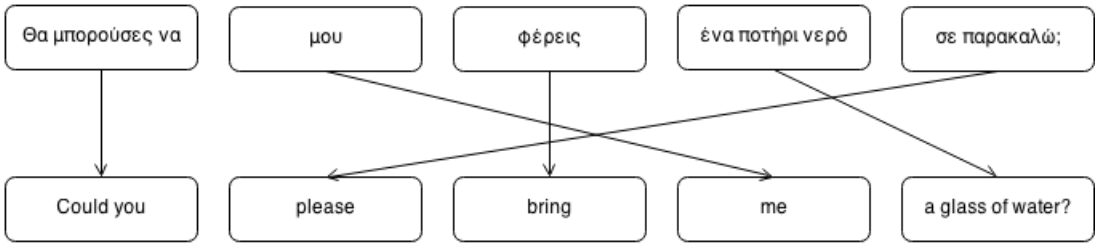

FIGURE 2.1: Example of phrase-based translation between Greek and English

2.3.3 Phrase-Based SMT

Until recently, the standard for SMT were word-based models, where the basic unit of the translation were the individual words. However, the current state of the art is phrase-based statistical machine translation. In a phrase-based system, the input sentence is segmented into phrases (units of one or more words) which are then translated separately in the target language and reordered, so as to produce the final output.

The use of phrases instead of words offers a way to tackle the problem that often there is not an one-to-one mapping between the words of the two languages. For example, in figure 2.1, there is no Greek word that corresponds to the English word "of". Two further advantages offered by phrase-based models (Koehn, 2010) are that they help in resolving translation ambiguities and that they make better use of the training data, as more training data result in more phrases learnt, something that might not stand for individual words.

The noisy-channel model that we previously showed for SMT is again the one that is used:

$$\hat{s} = arg\ max_s\ p(s)p(f|s)$$

with the difference that now the reverse translation probability of the whole sentence $p(f|s)$ is now further decomposed to the product of the translation probabilities of the individual phrases $\phi(f_i|s_i)$ and the reordering model. The reordering model can either be a simple cost function, eg. a distance-based cost function, or even better, a lexicalised one, also taking into account the probability that a particular phrase needs to be swapped or reordered.

Apart from the noisy-channel model, Och and Ney (2002) have suggested a more flixible translation model for phrase-based MT, using the maximum entropy framework. In this model, the posterior probability $p(s|f)$ is modelled directly as a combination of M feature functions h_m with λ weights:

$$\hat{s}^I = arg\ max_{s^I}\ \left[\Sigma_{m=1}^{M}\lambda_m h_m(s^I, f^J)\right]$$

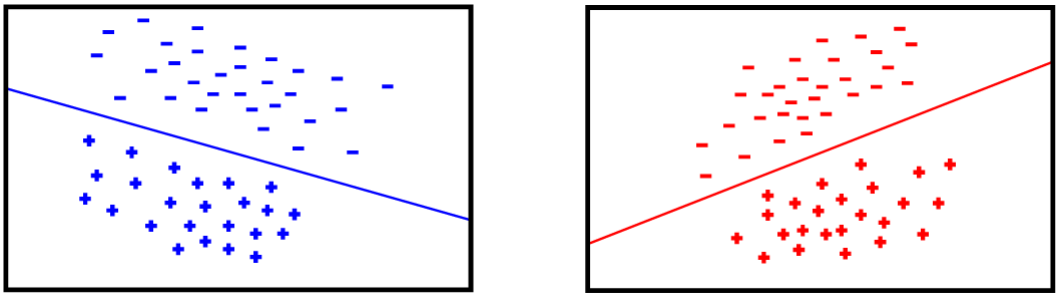

FIGURE 2.2: Two domains that require adaptation of a model.

where I, J are the number of phrases in s and f respectively. The advantage of this model is that enables an arbitrary number of features to be used, instead of just the reverse translation probability and the language model score. Other common additional features include translation probabilities, phrase and word counts, or phrase pair frequency.

2.3.4 Domain Adaptation for SMT

The task of domain adaptation is common for most problems that deal with natural language, as they are tackled as supervised learning problems, resulting into a kind of "overfitting" into the domain(s) of the training data. State-of-the-art performance is achieved for in-domain testing, but when the methods are applied to out-domain samples, the performance drops significantly. This is due to the fact that the parameters of the translation and language models reflect the empirical distribution of the training data domain, which can be very different from the distribution of the other domain.

For Language Processing and SMT, such domains can be the *news* domain, the *IT* or the *Legal* domain, or, more recently, the domains of Twitter, email or scientific articles. Each has its own characteristics, ranging from the language and the words used (for example, the *IT* domain is comprised of much more technical language) to the way the sentences/texts are structured (as, for example, in the *news* domain), or even other limitations, such as in sentence length (as Twitter only allows 140 characters).

An example that vividly illustrates the need for domain adaptation in a generic task is shown in figure 2.2. A model trained on one of the domains would perform badly on the other.

This can be due to several reasons:

- Different Distributions: The distribution of training and test data are different
- Unknown words: For a model trained in a specific domain, there are many unseen words in the new domain.

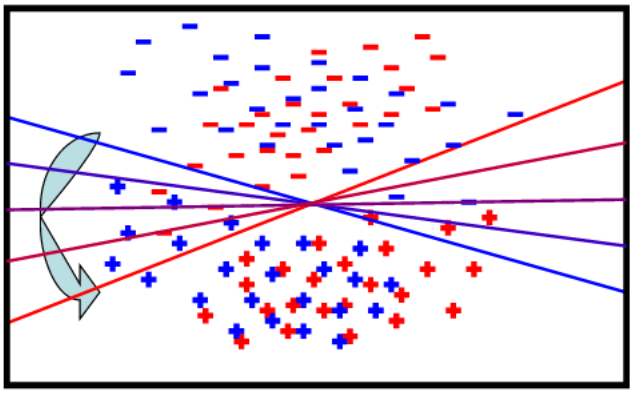

FIGURE 2.3: Adapting the model between two domains.

What is needed is a way to adapt the trained model so as to also fit the new domain, as figure 2.3 illustrates.

Several solutions have been suggested to tackle this problem that is evident in almost all Machine Learning tasks, according to the reasons that produce it:

- The problem of different distributions across the two domains can be tackled by weighting the instances accordingly (Bickel et al., 2007, Jiang and Zhai, 2007).
- Regularization was suggested to deal with the different labels of same instances, assuming that the labels across the domains are somehow close (Daumé III, 2007, Evgeniou and Pontil, 2004).

In the field of SMT, there has been intense interest in the task of domain adaptation, with several notable works, investigating the use of cheap monolingual resources (Bertoldi and Federico, 2009) or mixture modelling (Civera and Juan, 2007) to improve performance.

This interest in research for Domain Adaptation in SMT proves that indeed it is a problem that needs to be tackled, as it poses significant constraints over the performance of the SMT systems. However, the same applies to the QE task, since the differences on SMT performance due to domain change are also reflected in the quality of the MT output. Thus, Domain Adaptation is as needed for the QE task as it is for MT.

2.3.5 The CAT-tool scenario

An obvious application, witnessing the increasing adoption of MT technology, is aiding the work of professional (human) translators. This is called *Computer Assisted Translation (CAT)* and is done through dedicated software which are called CAT-tools.

Instead of humans translating sentences or whole documents from scratch, the CAT-tools provide them with translation suggestions, which are created using MT systems.

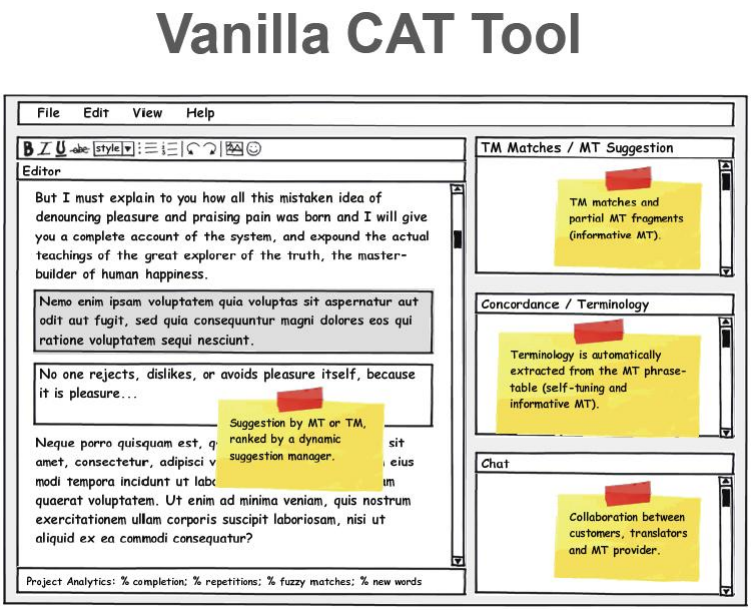

FIGURE 2.4: Example of a CAT-tool interface

Then the translators can post-edit the suggested translation, if needed, producing the final translation.

This process, post-editing MT output, is nowadays common practice among professional translators, since it increases their productivity. As estimated, the EU spends more than 1 billion € on translation costs per year. Semi-automating this process (as is the objective of the MateCat project - see §1.1.1) could lead to huge financial savings.

In a typical CAT-tool, the source/target text is split into segments and the translation progresses segment by segment. In addition, the CAT-tool provides help to the translators from different sources, like spell checkers, dictionaries, terminology managers, concordancers and *translation memory*. More recently, there is also research into using MT suggestions to aid the translators.

2.3.5.1 The Translation Memory (TM) Approach

Traditionally, CAT-tools were using *translation memories (TMs)*. All the previous work of the translator is stored in a database, called *translation memory*. The CAT-tool then, produces translation suggestions for the new instances by comparing them to the TM and finding perfect or fuzzy matches. The matches are then ranked according to the percentage of their match (100% matches are of course the top preference) and presented to the user as candidates for post-editing.

A TM can be shared among and simultaneously updated by several translators working on the same project. Thus, TM can model the style and terminology of a particular project or customer and help make a coherent and consistent translation.

In fact, professional translators had been relying on TMs long before SMT managed to produce reasonably good results. It is indeed helpful on a number of cases:

- on highly repetitive documents, like technical manuals or reports,
- on newer versions of previously translated documents, or
- when several translators work on the same project, ensuring consistency across the team.

Although TM can speed up the translation process in certain cases, in the general case the number of useful matches is rather small (5-10%), significantly limiting the helpfulness of TM.

2.3.5.2 Integrating Statistical MT into a CAT-tool

The advances on the field of Statistical Machine Translation have increased the interest in using SMT for a CAT-tool. For example, probably the most usually used among Google's special features is *Google Translate*, which provides translations for a large variety of language pairs for any input text. Instead of the TM, the translation suggestions towards the translators could be the outcome of an SMT system.

However, TMs are indeed quite useful. They, in fact, represent the translator's previous work and thus the fuzzy matches that produce translation suggestions are usually a good option for post-editing them and producing the final suggestion, at least in certain cases described in the previous section.

Thus, the creation of CAT-tools that use both SMT and TMs to produce translations has been suggested (He et al., 2010), in a framework where post-editors still work with TMs while benefiting from (better) SMT outputs. Deciding what the best outcome is, or what is the best way to combine the suggestions from SMT and TM, is still an open issue. Recent work (Federico et al., 2012), though, reports that such a system would benefit the translator's post-editing time, with the time gains being statistically significant for most translators that participated in the experiments.

2.4 MT Evaluation

The task of evaluating the quality of the machine translation output by using reference sentences is generally described as *MT Evaluation*.

2.4.1 Reference based MT Evaluation

The most commonly used metrics for SMT output quality evaluation are usually automatic ones, which compare the resulting translations with reference sentences.

Several metrics have been suggested, with the most notable being:

- BLEU (Papineni et al., 2002)) which currently represents the dominant evaluation metric
- NIST (Doddington, 2002)
- Meteor (Lavie and Agarwal, 2007)

All of these metrics measure the overlap of n-grams between the MT outputs and the reference sentences. BLEU uses a modified version of the standard machine learning *precision*, taking into account the times that an n-gram appears in the reference sentence. NIST is a modification of BLEU, also taking into account how informative each n-gram is. METEOR, on the other hand, is calculated by the harmonic mean of both precision and recall, giving 9 times more importance to recall.

The need, however, for reference sentences, makes these metrics quite costly to produce and are not actually usable in a real-life scenario. Furthermore, they have been shown to not correlate as well with human judgements at sentence level. In fact, results from Albrecht and Hwa (2007) indicate that high human-likeness does not imply good MT quality and vice-versa. Other metrics such as the one suggested by Specia et al. (2010), that are not based on reference-dependent features, but are rather based on the input and output sentences and (possibly) external corpora, have been shown to correlate much better with human evaluation.

2.4.2 Metrics of post-editing effort

The emergence of CAT-tools and the widespread application of SMT in the industry, has resulted in a growing interest on evaluating the MT output Quality, not compared to reference sentences (which in a real life scenario are not available), but in relation to the post-editing effort needed in order to produce the correct translation from the suggestion of the SMT system.

2.4.2.1 Human Annotations

One of the first metrics that were suggested was based on human annotations. Professional translators where presented with source sentences and the translation suggestion

and, after post-editing it in order to produce a publishable translation, they were asked to rate the post-editing effort according to the following scale:

- 1 = requires complete re-translation
- 2 = requires some re-translation, but post-editing is still quicker than re-translation
- 3 = very little post-editing needed
- 4 = fit for purpose.

Another suggested scale for the estimated effort is the following:

- 1 = The MT output is incomprehensible, with little or no information transferred accurately. It cannot be edited, needs to be translated from scratch.
- 2 = About 50-70% of the MT output needs to be edited. It requires a significant editing effort in order to reach publishable level.
- 3 = About 25-50% of the MT output needs to be edited. It contains different errors and mistranslations that need to be corrected.
- 4 = About 10-25% of the MT output needs to be edited. It is generally clear and intelligible.
- 5 = The MT output is perfectly clear and intelligible. It is not necessarily a perfect translation, but requires little or no editing.

2.4.2.2 Human Translation Edit Rate (HTER)

The *Human Translation Edit Rate (HTER)* has been introduced by (Snover et al., 2006) and it is a metric of the distance between the suggested translation and the final post-edited version.

It is computed as:

$$HTER = \frac{\#edits}{\#postedited\ words}$$

where the number of edits include insertion, deletion and substitution of single words, and the shifting of word sequences.

The result is a continuous score in $[0, 1]$, where 0 denotes an excellent suggested translation which didn't need any alternation and 1 denotes a translation of very low quality which, in fact, needed to be rewritten from scratch, as the number of the edits is at least equal to the number of the words.

For example, given this *source* sentence

```
Because I also have a penchant for tradition , manners and customs .
```

the produced spanish translation from the SMT system:

```
Porque también tengo una inclinación por tradición , modales y
costumbres .
```

and the *post-edited* sentence:

```
Porque también tengo una inclinación por la tradición , los modales y
las costumbres .
```

the HTER score, because the number of edits is 3 (insertions of the words `la, los` and `las`) and the number of the post-edited words equals the length of the *post-edited* sentence of 15, is

$$HTER = \frac{3}{15} = 0.20$$

2.4.2.3 Post-editing Time

Another suggested measure of post-editing effort was the average number of seconds needed to post-edit each word in a sentence[1]. For a sentence s with n words that required t seconds to post-edit, this is computed as:

$$time = \frac{t}{n}$$

It is considered that a low time needed for post-editing indicates a good translation, whereas larger time indicates a bad translation. However, the measured time usually includes:

- *reading* time,
- *searching* for information on external sources,
- *typing* time,
- *extra* time for any secondary activity (e.g. correction).

All these variables might significantly vary across sentences and translators, thus this metric includes high variability and "noise".

[1] suggested in (Specia, 2011) and further examined in(Koponen et al., 2012)

2.5 Quality Estimation

Although the current standard is reference-based MT evaluation, the ultimate goal is to be able to *predict* the quality of the MT output at run-time and without reference translations (Blatz et al., 2003, Specia et al., 2009). This task, consisting in estimating the quality of a system's output for a given input, without information about the expected output, is called *Quality Estimation (QE)*.

Traditionally, QE was viewed as a binary classification problem, where the task was to decide whether the output was relatively "good" or "bad". However, the boundary between what could be called a "good" or a "bad" translation is rather blurry, not to mention the fact that such information might be useless in certain applications, such as post-editing MT output, where one needs an estimation of the post-editing effort and not of the quality of the translation.

The first attempts on QE for MT aimed at estimating the quality at word or phrase level. Starting with (Blatz et al., 2004), QE at sentence level used regressors and classifiers trained on extracted features and labels from MT metrics like NIST. For classification, the threshold for "good" translations was set at the 5^{th} or 30^{th} percentile of the translations' NIST scores.

Quirk (2004) also used classification and a pre-defined threshold, building upon manually labelled data for quality, which outperformed models trained on even larger, *automatically* annotated though, data. On the other hand, Gamon et al. (2005) focused more on *human-likeness classification*, using linguistic features and trying to distinguish between machine- or human-generated translations.

Finally, Specia et al. (2010) extend QE from a binary classification task and propose that it can be seen as a regression task, producing a score (either continuous or discrete) in a given range. They also suggest that such a score, based on a number of features extracted from the source/target sentences and/or monolingual or parallel corpora, would me more valuable for practical applications like filtering out bad translations for human post-editing. They show, finally, that such a score would be more suitable than reference-based metrics for MT evaluation for certain tasks like selecting the best translation from a collection of MT systems to present to a user.

2.5.1 Comparison of the suggested metrics

Although any of the non-reference-based metrics for MT evaluation would be suitable for Quality Estimation, there are certain advantages or disadvantages associated with each one's use.

It has been shown (Koponen, 2012) that human annotations of post-editing effort, which reflect the translators' perception, do not always correlate well with edit distance metrics

such as HTER. This means (Koponen et al., 2012) that technical and cognitive effort are not always equal.

In addition, although the metric of post-editing *time* has been advocated as the best metric (Koponen et al., 2012), or used (Specia, 2011) in experiments, it is highly unreliable. It might vary among post-editors, based on their experience, knowledge of the translation subject/domain, it might even vary due to various unexpected - and uncountable - factors (eg. someone interrupting the translator while working).

To conclude, throughout this thesis we will focus only on QE (and not MT evaluation) and we will use *HTER* as the measure of post-editing effort, because, by only relying on the data, it is more coherent that the other metrics that involve individual opinions (*human annotations*) or unpredictable factors (*time*).

2.6 Related Work

2.6.1 Evaluation Campaigns

Throughout the last years there have been evaluation campaigns in the form of shared tasks in major conferences. The most important is held once per year, at the Workshop on Machine Translation (WMT) which, in the last years, is hosted by the Annual Conference of the Association of Computational Linguistics (ACL).

These shared tasks focus on different areas of research on MT:

- Performance of SMT systems on the Translation task
- Performance of SMT systems on featured translation tasks (eg. WMT11 (Callison-Burch et al., 2011) featuring English-Haitian Creole translation)
- Performance on ranking translations of sentences
- Deciding on appropriate evaluation metrics

In the last two years, one of the shared tasks was dedicated on Quality Estimation. The results of these shared tasks are presented in the next section.

2.6.1.1 WMT 12 Quality Estimation Shared Task

In the first year that the Shared Task on Quality Estimation was introduced (Bojar et al., 2013), its goals were to:

- identify new and effective quality indicators (features)

- identify alternative machine learning techniques for the problem
- test the suitability of the proposed evaluation metrics for quality estimation systems
- establish the state of the art performance in the field
- contrast the performance of regression and ranking techniques

It featured two subtasks, one for predicting the score of QE and another for ranking instances according the QE. 11 teams submitted systems for both tasks, with 5 of them performing above the very competitive baseline system used for comparison. The weighted average of 3 human annotations was used as the training label and the predicted value should be a value in the [1,5] range, depicting the post-editing effort according to the scale described in §2.4.2.1.

The metrics for evaluating the performance of systems were the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE), which are defined in §2.6.1.3.

2.6.1.2 WMT 13 Quality Estimation Shared Task

Given the experience obtained from the previous year, the QE shared task of WMT13 featured wider coverage of the issues of QE (Bojar et al., 2013). To begin with, it didn't focus only on sentence-level QE, but also on word-level QE.

Suggested probable uses of word-level QE are:

- Highlight words that need editing in post-editing tasks.
- Inform readers/translators for portions of the text that might not be reliable.
- Select the best segments among multiple options coming from various SMT systems, eg. for the creation of an MT system from other MT systems combinations.

So, the goals of this shared task were to:

- explore various granularity levels for the task (sentence-level and word-level).
- explore the prediction of more objective scores such as edit distance and post-editing time.
- explore the use of quality estimation techniques to replace reference-based MT evaluation metrics in the task of ranking alternative translations generated by different MT systems.

- identify new and effective quality indicators (features) for all variants of the quality estimation task.
- identify effective machine learning techniques for all variants of the quality estimation task.
- establish the state of the art performance in the field.

It featured 4 subtasks. Three of them focused on sentence-level QE:

- Predicting post-editing distance in terms of *HTER*
- Selecting the best translation for post-editing
- Predicting post-editing time

and the other was dedicated on classification for word-level QE.

For the scoring variant of the post-editing distance prediction, 9 out of the 16 submissions performed significantly better than the baseline system, with the results being in general better than the previous year's results. Again, the metrics for evaluating the performance of systems were the MAE and RMSE.

2.6.1.3 Evaluated Systems

In this section the best performing systems of the WMT12 and WMT13 QE shared tasks are presented.

The metrics for evaluating the performance of the systems (which produced y_i predictions over n sentences with $\hat{y}_i$ true labels) were the Mean Absolute Error (MAE)

$$MAE = \frac{\Sigma_{i=1}^{n}|\hat{y}_i - y_i|}{n}$$

and the Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

In both the tasks, a baseline system was developed. It used a feature extraction software which provided 17 *baseline* features (the baseline features are more thoroughly described in §3.5.1). These features were used to train an SVM regression algorithm with RBF kernel, using the LIBSVM package (Chang and Lin, 2011) in WMT12 and the Scikit-learn toolkit (Pedregosa et al., 2011) in WMT13, having optimized parameters through grid-search and 5-fold cross-validation.

System ID	MAE	RMSE
SDLLW_M5PbestDeltaAvg	0.61	0.75
UU_best	0.64	0.79
SDLLW_SVM	0.64	0.78
UU_bltk	0.64	0.79
Loria_SVMlinear	0.68	0.82

TABLE 2.1: Best performing systems at WMT 12 Quality Estimation Shared Task

System ID	MAE	RMSE
SHEF FS	12.42	15.74
SHEF FS-AL	13.02	17.03
CNGL SVRPLS	13.26	16.82
LIMSI	13.32	17.22
DCU-SYMC combine	13.45	16.64

TABLE 2.2: Best performing systems at WMT 13 Quality Estimation Shared Task

The best performing systems in each year's task for predicting a QE score (1-5 scale for WMT12 and HTER score for WMT13) are presented in tables 2.1 and 2.2.

It is worth noting the difference in the MAE between the two years, which is the result of using human annotations in a 1-5 scale (WMT12) and using HTER (WMT13).

The main characteristics of the best-performing systems were:

- WMT12:
  - ***SDLLW*:** This system, which was ranked as *best* in the WMT12 workshop, was developed by the SDL Language Weaver (USA, (Soricut et al., 2012)). It uses three sets of features
    - the 17 baseline features
    - 8 system-dependent features from the decoder logs of Moses
    - 20 features developed internally

    and feature-selection algorithms that optimize towards the relevant metric for each task. This feature-selection process, though, is computationally intensive. The $\_M5PbestDeltaAvg$ variant uses a resulting 15 features set and builds a decision tree using an M5P model. The $\_SVM$ variant uses a resulting 20 feature set to build an SVM epsilon regression model with RBF kernel.
  - ***UU*:** Developed by the Uppsala University (Sweden, (Hardmeier et al., 2012)) this system uses the 17 baseline features, 82 additional features (Hardmeier, 2011)) and information from constituency and dependency trees, extracted from the Stanford and Malt Parser respectively. The models for both variants are SMV regression models with polynomial kernels.

  - ***Loria_SVMlinear*:** This system, developed at LORIA Institute (France, (David et al., 2012)), builds SVM regression models with linear kernel, using feature selection over the 17 baseline features and 49 additional features (Raybaud et al., 2011).

From all the systems that achieved good results in the WMT12 shared task, it is worth noting the importance of feature selection. Starting from a large set of features, extracted with various tools, the top 5 systems employed some kind of feature-selection in order to identify the more appropriate features for the task.

- WMT13:
  - ***SHEF*:** The two submissions that were ranked in the two top places at the WMT13 workshop, developed at University of Sheffield, (UK (Beck et al., 2013)) are the current state-of-the-art at QE. These systems use Gaussian Processes along with feature selection using optimizing hyperparameters and active learning that is used to reduce the training size. The features are selected by their relevance for the model by the gaussian process algorithm itself. The query selection strategy for active learning is based on the informativeness of the features.
  - ***CNGL SVRPLS*:** This system was developed at the Centre for Next Generation Localization (Ireland (Bicici, 2013)). It builds SVR models with Partial Least Squares with features selected in such a way (MTPP (Biçici et al., 2013)) so as to enable language independent and MT system independent predictions.
  - ***LIMSI*:** Developed by the Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur, (France (Singh et al., 2013)), this system uses simple elastic regression. However, it employs features that are somehow not-traditional. Several features are based on large span continuous space language models; apart from that, though, the features are always calculated in normalised forms against both the source and target sentences length, or in a ratio form, giving the ratio of the feature when calculated against the source sentence and when calculated against the target sentence.
  - ***DCY-SYMC combine*:** This system is the result of the collaboration of the University of Dublin and Symantec (Ireland, (Rubino et al., 2013)) and it uses SVR models, combined with regression tree models for feature selection, which reduce the initial set of 442 features (extracted with several NLP tools) to 134.

Again, the focus of the attention was the struggle to define the features that are more appropriate for the QE task. The features and the feature selection methods are getting more "fancy" or "complex". In addition the issue of the learning methods that should be used is investigated, with the best performing system diverging from the "norm" of SVM regression and employing Gaussian Processes.

2.7 Challenges

Despite the substantial progress done so far in the field, also boosted by the previously mentioned successful evaluation campaigns, focusing on concrete market needs makes possible to further define the scope of research on QE.

For instance, moving from the controlled lab testing scenario to a real working environment poses additional constraints in terms of adaptability of the QE models to the variable conditions of a translation job. Systems' capability to self-adapt to the behaviour of specific users and domain changes are facets of the problem that so far have been disregarded. Current systems, typically designed for the controlled setting of QE shared tasks, are in fact *static* and optimized for datasets where training and test instances reflect similar distributions. Besides, in the controlled experiment environment, the *train* and *test* are available in advance.

Ideally, instead, the CAT-tools should confront with the fact that:

1. **The notion of MT output quality is highly subjective** (Koponen, 2012, Turchi et al., 2013). Since the quality standards of individual users may vary considerably (*e.g.* according to their knowledge of the source and target languages), the estimates of a *static* model trained with data collected from a group of post editors might not fit with the actual judgements of a new user.

2. **Each translation job has its own specificities** (domain, complexity of the source text, average target quality). Since the data from a new job may differ from those used to train the QE component, its estimates on the new instances might result to be biased or uninformative.

Chapter 3

Adaptive Quality Estimation

3.1 Introduction

In order to respond to the challenges mentioned in the conclusion of the previous chapter, one possible direction would be to create an *adaptive* Quality Estimation system. This means moving from the current trend of tackling Quality Estimaion as a *supervised machine learning regression* task with *batch* methods, to employing *online* (or *adaptive*) learning methods.

On the MT system side, research on adaptive approaches tailored to interactive SMT and CAT scenarios explored the online learning protocol (Littlestone, 1988) to improve various aspects of the decoding process (Bertoldi et al. (2013), Cesa-Bianchi et al. (2008), Martínez-Gómez et al. (2011, 2012), Mathur et al. (2013), Ortiz-Martínez et al. (2010)).

However, as regards QE models, so far there has been no investigation on incremental adaptation by exploiting users' feedback to provide targeted (system, user, domain or project specific) quality judgements.

Incremental methods should be able to adapt to the changes that result from varying the post-editor or the translation domain across translation jobs, resulting in better performance than the currently employed *batch* methods, thus motivating the use of the online framework and specific, adaptive algorithms.

In the remainder of this section, the basic concepts of machine learning are introduced. In addition, algorithms for adaptive learning are described, along with the framework through which they can be applied to the QE task.

3.2 Machine Learning

3.2.1 Definition

Machine Learning as defined by A.L. Samuel (Samuel, 2000) is the field of study that gives computers the ability to learn without being explicitly programmed.

A Learning problem can be defined (Mitchell, 1999) as follows:

> A computer program is said to *learn* from experience e with respect to some task t and some performance measure p, if its performance on t, as measured by p, improves with experience e.

Typical fields in which Machine Learning has been used are Natural Language Processing (NLP), Pattern Recognition, Computer Vision. In addition, it is now widely adopted by the commercial sector, being useful in applications such as database mining, spam detection, or self-customizing programs.

3.2.2 Machine Learning Process

Machine learning algorithms can be divided into these main categories (Haykin et al., 2009):

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- others, like semi-supervised learning, recommender systems, etc.

Supervised Learning

Supervised Learning is a technique trying to infer a function from labeled training data. A supervised learning algorithm receives data pairs of input (typically a vector) and desired output. Based on the input, it infers a function which can be used to map new examples. The output of the function can be either a continuous value (regression task) or a discrete value/class label of the input (classification task).

Unsupervised Learning

Unsupervised Learning, in contrast to *supervised learning*, does not receive labeled data, but only uses unlabeled input, trying to infer a "hidden" structure of the input, in a self-organising way. Typical unsupervised learning tasks include clustering, social network analysis or astronomical data analysis.

Reinforcement Learning

Reinforcement Learning is concerned with how *agents* in certain *states* ought to behave in certain *environments* in order to maximize some notion of cumulative *reward*.

3.3 Adaptive (online) Machine Learning

Adaptive Learning, or *online learning*, is a way of performing *supervised learning* that takes place in a sequence of consecutive rounds. In each round, the model is given a question and is required to provide an answer to this question.

This answer can be as simple as a yes/no decision, as in the case of binary classification, or as complex as a string or a real number, as in the case of regression.

To answer the question, the model uses a prediction mechanism, a hypothesis, which actually is a mapping from the set of questions to the set of admissible answers. After predicting an answer, the algorithm receives feedback indicating the correct answer and uses it to modify the prediction mechanism, if it is needed.

The model is updated according to the quality of the answer that it previously gave. The quality of the answer is assessed by a loss function, which measures the discrepancy between the prediction and the correct answer.

The goal is for the model to, ultimately, minimize the cumulative loss suffered along its run, by using the feedback of every round, so that it will be more accurate in the next rounds.

3.4 Online Learning Algorithms

This section is dedicated to the description of the *online learning* algorithms that we are using and the theory that lies behind them.

3.4.1 Online Support Vector Regression (OnlineSVR)

3.4.1.1 Support Vector Machines

Support Vector Machines (SVM) were formulated by Vapnik and although they were used initially for classification, they have been adapted for use in the regression task (Vapnik et al., 1997).

The basic idea, for classification, was that

> given a training set, the support vector machine creates a hyperplane as a decision rule, in a way that the distance between the training points of either class and the hyperplane (called margin) is maximised.

3.4.1.2 Support Vector Regression

Support Vector Regression (SVR) is the adaptation of SVMs for the regression task. The main idea is the same (creating a hyperplane which maximizes the margin) but this time an error tolerance value ϵ is introduced.

Given a training set $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \subset X \times \Re$ of n training points, were x_i is a vector of dimensionality d (so $X = \Re^d$), and $y_i \in \Re$ is the target, the goal is to find a hyperplane (or function) $f(x)$ that has at most ϵ deviation from the target y_i, and at the same time it is as flat as possible.

The solution now, instead of a line (in the case of $d = 1$), is an ϵ-tolerant to errors "tube", as shown in figure 3.1. In the general case ($d > 0, d \in \mathbb{Z}$), instead of a hyperplane (which was created for classification), the solution is a hyper-slab of 2ϵ width.

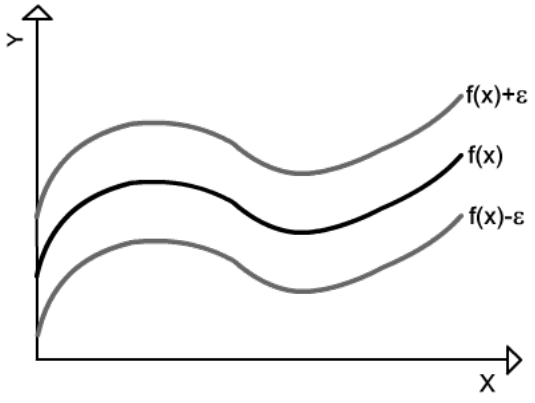

FIGURE 3.1: The SVR tube

Note that, in the high dimensionality setting, the data might not be linearly separable. The easiest way to deal with this problem is to pre-process the *training* data and map them to a feature space **F**.

In the end, given the training set mentioned before, the linear regression function is

$$f(\mathbf{x}) = \mathbf{W}^T \Phi(\mathbf{x}) + b$$

on a feature space $\mathbf{F}$, where $\mathbf{W}$ is a vector in $\mathbf{F}$ and $\Phi(\mathbf{x})$ maps the input $\mathbf{x}$ to a vector in $\mathbf{F}$. This can be written formally as a convex optimization problem by requiring:

$$\begin{aligned} &\text{minimize } \frac{1}{2}\|\mathbf{W}\|^2 \\ &\textit{subject to} \begin{cases} y_i - \mathbf{W}^T\Phi(\mathbf{x}) - b & \leq \epsilon \\ \mathbf{W}^T\Phi(\mathbf{x}) + b - y_i & \leq \epsilon \end{cases} \end{aligned}$$

The dual optimization problem based on Lagrange multipliers, after substituting for the condition that the partial derivatives of the dual problem with respect to the primal variables have to vanish for optimality, can be written as follows (Smola and Schölkopf, 2004):

$$\begin{aligned} &\text{maximize } \left\{ -\frac{1}{2}\Sigma_{i,j=1}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle - \epsilon\Sigma_{i=1} y_i(\alpha_i - \alpha_i^*) \right\} \\ &\textit{subject to } \Sigma_{i=1}(\alpha_i - \alpha_i^*) = 0 \textit{ and } \alpha_i, \alpha_i^* \in [0, C] \end{aligned}$$

where α_i, α_i^* are Langrange multipliers.

Now finding the optimal values for $\mathbf{W}$ and b is done by solving this dual optimization problem by utilizing the **Karush-Kuhn-Tacker (KKT)** conditions. These conditions state that at the optimal solution the product between dual variables and constraints has to vanish. The final solution is easily found by using Mercer (Mercer, 1909) kernels:

$$f(\mathbf{x}) = \Sigma_{i=1}^{N}(\alpha_i^* - \alpha_i)k(x_i, x) + b$$

The most common kernels ($k(x_i, x)$) used are:

- Linear kernel
- Polynomial kernel
- RBF kernel
- Gaussian kernel

3.4.1.3 Online Support Vector Regression

The previously described theory of SVM and SVR was only used in a *batch* fashion. That means that the solution to the optimization problem was found after taking all

training instances into account. Building upon the theory of ϵ-insensitive SVR, Ma et al (Ma et al., 2003) described an incremental algorithm for Accurate Online SVR, which did not require re-training over the whole training set if a new instance was to be added.

The incremental algorithm updates the trained SVR function for each sample that is added (in an online fashion this time) to the *training* set. The details are skipped, but

> "...the main idea is to update the coefficient θ_c of the margin of the new sample x_c in a finite number of steps until it meets the KKT conditions. In the same time, though, it must be ensured that also the rest of the existing samples continue to satisfy the KKT conditions."

Solving the simplified optimization problem as before, we end up with three sets into which the *training* samples are classified in each iteration:

- the E set of the Error Support Vectors, for which the margin is greater than ϵ
- the S set of the Margin Support Vectors, for which the margin is exactly ϵ
- the R set of the Remaining Samples, for which the margin is less than ϵ.

3.4.2 Passive Aggressive Algorithms (PA)

Passive Aggressive (PA) Algorithms were presented at (Cramer et al., 2007). They are based on the same idea as SVMs, using hypotheses from the set of linear predictors.

Again, each instance is represented by a feature vector. The prediction mechanism is based on an ϵ-insensitive loss function, meaning that it creates a hyper-slab of width 2ϵ dividing the instance space, and the *margin* of an instance is its distance to the borders of the hyperplane.

The difference with SVMs lies in the way that the incrementally learned vector, which describes the hyper-slab, is updated.

The loss that is suffered is ϵ-insensitive. This means that for a margin up to ϵ, the loss is 0. Thus, the loss function now used is the ϵ-insensitive loss function

$$l_\epsilon \mathbf{W}; (\mathbf{x}, y) = \begin{cases} 0, & if\ |\mathbf{W} \cdot \mathbf{x} - y| \leq \epsilon \\ |\mathbf{W} \cdot \mathbf{x} - y| - \epsilon, & otherwise \end{cases}$$

where $\mathbf{W}$ is the incrementally learned vector, $\mathbf{x} \in \Re^d$ is an instance of dimensionality d and y is its respective target.

The result of the loss function is used to update the model. If the loss for a certain instance $\mathbf{x}_t$ is zero, then the algorithm reacts *passively* and does not update the model:

$$\mathbf{W}_{t+1} = \mathbf{W}_t.$$

However, if the loss for an instance $\mathbf{x}_t$ is not zero, the algorithm reacts *aggressively* and forces the model to update in order to satisfy the constraint $l(\mathbf{W}_{t+1}, (\mathbf{x}_t, y_t)) = 0$. The solution to this optimization problem is simple:

$$\mathbf{W}_{t+1} = \mathbf{W}_t + sign(y_t - \hat{y}_t)T_t\mathbf{x}_t,$$

where $T_t = \frac{l_t}{\|\mathbf{x}_t\|^2}$ is the update needed to satisfy the constraint.

However, in certain cases this aggressive update for every instance might completely "confuse" the model, resulting in worse performance, in case, e.g. an outlier is present in the set of the instances. To avoid such problems, an *aggressiveness* parameter C is introduced, signifying the maximum allowed update.

In the end, the process for the *PA* algorithms is the following:

1. initialize aggressiveness parameter $C > 0$
2. initialize tolerance parameter $\epsilon > 0$
3. initialize vector $\mathbf{W}_1 = (0, 0, ..., 0)$
4. For $t = 1, 2, ...$:
   (a) Receive instance $\mathbf{x}_t \in \Re^n$
   (b) Predict value $\hat{y}_t = \mathbf{W}_t \cdot \mathbf{x}_t$
   (c) Receive correct label $y_t \in \Re$
   (d) Suffer ϵ-insensitive loss: $l_\epsilon \mathbf{W}_t; (\mathbf{x}_t, y_t) = max(0, |\mathbf{W}_t \cdot \mathbf{x}_t - y_t| - \epsilon)$
   (e) Set update $T_t = min(C, \frac{l_t}{\|\mathbf{x}_t\|^2})$
   (f) Update model: $\mathbf{W}_{t+1} = \mathbf{W}_t + sign(y_t - \hat{y}_t)T_t\mathbf{x}_t$

3.4.3 Online Gaussian Process (OnlineGP)

3.4.3.1 Gaussian Processes

Rasmussen (2006) and Williams and Rasmussen (1996) define a *Gaussian Process (GP)* as *"a collection of random variables, any finite number of which have a joint Gaussian*

distribution" and show that any Gaussian Process can be completely defined by its mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(x)$:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))\right] \end{aligned}$$

As a result, the Gaussian Process can be written as:

$$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

Again, denoting $\mathbf{x} \in \Re^d$ as an instance of dimensionality d and y its respective target, we consider that the whole data set consists of n pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$.

The Gaussian Process assumes that every target y_i is generated from the corresponding data $\mathbf{x}_i$ and an added white noise η as:

$$y_i = f(\mathbf{x}_i) + \eta, \quad where \quad \eta \sim \mathcal{N}(0, \sigma_n^2)$$

This function $f(\mathbf{x})$ is drawn from a GP prior:

$$f(x) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

where the covariance is encoded using the kernel function $k(\mathbf{x}, \mathbf{x}')$.

In addition, the kernel function can be modified so as to enable feature selection. For example for an RBF kernel, the kernel function can incorporate a matrix A:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \; exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T A^{-1}(\mathbf{x} - \mathbf{x}')\right)$$

This matrix A is diagonal and encodes the smoothness of functions f with respect to each feature. This means that, for unimportant features the functions are relatively flat, whereas for important features the functions are more jagged. The values for A are learned automatically from the data, creating a so-called *automatic relevance determination (ARD)* kernel.

In the end, the prediction y_i for an instance $\mathbf{x}_i$ can be found from:

$$p(y_i|\mathbf{x}_i, \mathcal{D}) = \int_f p(y_i|\mathbf{x}_i, f)p(f|\mathcal{D})$$

which, when solved analytically, gives:

$$y_i \sim \mathcal{N}\left(\mathbf{k}_i^T(\mathcal{K} + \sigma_n^2 I)^{-1} y, \mathbf{k}(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{k}_i^T(\mathcal{K} + \sigma_n^2 I)^{-1}\mathbf{k}_i\right)$$

where $\mathbf{k}_i = [k(\mathbf{x}_i, \mathbf{x}_1), k(\mathbf{x}_i, \mathbf{x}_2), ..., k(\mathbf{x}_i, \mathbf{x}_n)]^T$ are the kernel evaluations between the test point and the training set, and $\mathcal{K}$ is the kernel matrix over the training points with values $\mathcal{K}_{ij} = k(x_i, x_j)$.

Note the posterior $p(f|\mathcal{D})$ which reflects a belief, updated according to the training set, over the possible functions f.

GPs are considered the state-of-the-art for regression in general, because:

- they are probabilistic models and support Bayesian inference (as compared to SVMs that are not probabilistic)
- they provide greater flexibility in fitting the kernel hyperparameters even in complex cases

Finally, Cohn and Specia (2013) and Beck et al. (2013) have introduced Multi-Task Gaussian Models, which enable for learning from multiple labels over the same instances. This can prove quite useful for modelling multiple different post-editors.

3.4.3.2 Online Gaussian Processes (Online GPs)

Csató and Opper (2002) introduced the online version of gaussian processes. As described in the previous section, in the GP framework, the parameters that are learnt are functions and the GP priors specify a Gaussian distribution over a function space.

Thus, for $\mathbf{f} = \{f(x_1), ..., f(x_N)\}$ a set of functions values such that $\mathbf{f}_D \subseteq \mathbf{f}$, where $\mathbf{f}_D$ is the set of $f(x_i) = f_i$ with x_i in the observed set of inputs, the posterior distribution can be computed by using the data likelihood together with the prior $p_0(\mathbf{f})$ as:

$$p_{post}(\mathbf{f}) = \frac{P(\mathcal{D}|\mathbf{f})p_0(\mathbf{f})}{\langle P(\mathcal{D}|\mathbf{f}_\mathcal{D})\rangle_0}$$

where $\langle P(\mathcal{D}|\mathbf{f}_\mathcal{D})\rangle_0$ is the average of the likelihood with respect to the prior GP, which for online learning is GP at time $\prime$.

Following simple properties of Gaussian distribution, Csató and Opper (2002) proves that the result of a bayesian update using a GP prior with mean function $\langle \mathbf{f}_x \rangle_0$, kernel $K(x, x')$ and data $\mathcal{D} = \{(x_i, y_i)|i = 1, ..., N\}$, is a process with mean and kernel function given by:

$$\langle \mathbf{f}_x \rangle_{post} = \langle \mathbf{f}_x \rangle_0 + \Sigma_{i=1}^N K_0(x, x')q(i)$$

$$K_{post}(x, x') = K_0(x, x') + \Sigma_{i=1}^N \mathrm{K}_0(x, x')R(ij)K_0(x_j, x').$$

where the parameters $q(i), R(ij)$ only have to be computed once during training and can be used when making predictions. However, this representation is immediately helpful,

because the posterior process is usually not Gaussian. It is therefore needed to approximate the process with a Gaussian one. This approximation is computed by trying to minimize the Kullback-Leibler divergence between the true and the approximate distribution. The Kullback-Leibler divergence between two distributions p, q with densities $p(\theta), q(\theta)$ is defined as:

$$KL(p,q) = \int p(\theta) ln \frac{p(\theta)}{q(\theta)} d\theta$$

Assuming, now, that we have computed the Gaussian approximation $\hat{p}_t$ after t training samples, the updated posterior can be computed using the Bayesian rule:

$$p_{post}(\mathbf{f}) = \frac{P(y_{t+1}|\mathbf{f})\hat{p}_t(\mathbf{f})}{\langle P(y_{t+1}|\mathbf{f}_{\mathcal{D}})\rangle_t}$$

Now it is fairly easy to project this non-Gaussian posterior to a Gaussian approximation $\hat{p}_{t+1}$ by minimizing the divergence $KL(p_{post}|\hat{p}_{t+1})$, as the posterior only contains the likelihood of one sample. Using again parametrisation, the mean and kernel of the update after this step will be:

$$\langle f_x \rangle_{t+1} = \langle f_x \rangle_t + q^{t+1} K_t(x, x_{t+1})$$

$$K_{t+1}(x, x') = K_t(x, x') + r^{t+1} K_t(x, x_{t+1}) K_t(x_{t+1}, x')$$

where q^{t+1} and q^{t+1} are again easily computed.

One interesting variant suggested by Csató and Opper (2002), is the one where the update is only performed when a certain measure for the approximation error is not exceeded. A *Basis Vector set* $\mathcal{BV}$ with pre-defined capacity will store the set of inputs for which the *exact* update is performed and for which the number of parameters is increased. As samples come in an online fashion, some will be included in the $\mathcal{BV}$ and others will be left out. If the $\mathcal{BV}$ is full and a new sample is found to be important, the BV vector with the smallest error is removed from $\mathcal{BV}$ and is replaced, in an online fashion, by the new input vector.

Thus, unfolding the recursion over the recursion steps in the update rules, the posterior GP can be written only in terms of the $\mathcal{BV}$ and the initial kernel and likelihoods:

$$\langle f_x \rangle = \Sigma_{i \in \mathcal{BV}} K_0(x, x_i) a(i)$$

$$K_0(x, x') = K_0(x, x') + \Sigma_{i,j \in \mathcal{BV}} K_0(x, x_i) C(ij) K_o(x_j, x')$$

where the parameters a, C and an extra parameter s (introduced for clarity) are computed in an online fashion:

$$a_{t+1} = T_{t+1}(a_t) + q^{t+1} s_{t+1}$$

$$C_{t+1} = U_{t+1}(C_t) + r^{t+1} s_{t+1} s_{t+1}^T$$

$$s_{t+1} = T_{t+1}(C_t k_{t+1}) + e_{t+1}$$

where $k_{t+1} = k_{x_{t+1}}$ and e_{t+1} is the $(t+1)$-th unit vector.

The error of the approximation at each step can be computed as

$$\epsilon_{t+1} = |q^{t=1}|\gamma_{t+1}$$

where

$$\gamma_{t+1} = k^*_{t+1} - \mathbf{k}^T_{t+1}\mathbf{K}^{-1}_t\mathbf{k}_{t+1}$$

In order to reduce computational complexity, the inverse Gram matrix $\mathbf{Q}_t = K_t^{-1}$ is also kept and updated in an online fashion:

$$\mathbf{Q}_{t+1} = U_{t+1}(\mathbf{Q}_t) + \gamma^{-1}_{t+1}(T_{t+1}(\hat{e}_{t+1}) - e_{t+1})(T_{t+1}(\hat{e}_{t+1}) - e_{t+1})^T.$$

The final algorithm is initializing $\mathcal{BV}$ as an empty set with maximal number of elements d, a prior kernel K_0 and a tolerance parameter ϵ_{tol}. The GP parameters a, C and the inverse Gram matrix Q are set to empty values.

Then, for each iteration for data (x_{t+1}, y_{t+1}):

1. Compute $q^{t+1}, r^{t+1}, k^*_{t+1}, \mathbf{k}_{t+1}, \hat{e}_{t+1}$ and γ_{t+1}
2. If $\gamma_{t+1} < \epsilon_{tol}$ then perform a reduced update (without adding the sample to the $\mathcal{BV}$ or extending the size of $\mathbf{a}$ or $\mathbf{C}$
3. Else perform full update by adding the current input to the $\mathcal{BV}$ set computing the update for the inverse Gram matrix $\mathbf{Q}$
4. If the size of $\mathcal{BV}$ exceeds d, then compute the scores ϵ_i for all vectors in the $\mathcal{BV}$ and delete the one with the minimum score.

3.5 Applying Online Learning Methods to the QE task

The QE task is generally treated as a supervised learning regression task. As shown in section §2.5 it has been so far treated in the batch learning mode.

This section provides the framework through which *online* learning methods can be applied to the QE task, firstly in a general overview and secondly in the CAT-tool environment.

3.5.1 General Framework

When dealing with Quality Estimation, all we have is $[src, trg, pe]$ tuples. However, a supervised machine learning regression task requires $(\mathbf{x}, y)$ tuples, where $\mathbf{x}$ is a vector with features describing the input and y is the desired output.

Thus, what is needed is a mapping of $[src, trg, pe]$ tuples to $(\mathbf{x}, y)$ tuples.

As input is considered the $[src, trg]$ tuple. The answer to transforming it into a $\mathbf{x}$ vector is using features extracted from this tuple as values for the vector.

Suggested (Specia et al., 2009, 2010) features are:

- source and target sentence length (in tokens)
- ratio of source and target sentence length
- source and target sentence 3-gram language model probabilities and perplexities
- source and target sentence type/token ration
- average source word length
- percentage of 1 to 3-grams in the source sentence belonging to each frequency quartile of a monolingual corpus
- number of mismatching opening/closing brackets and quotation marks in the target sentence
- number of punctuation marks in the source and target sentences
- average number of occurrences of all target words within the target sentence
- average number of translations per source word in the sentence (as given by IBM 1 table thresholded so that $prob(t|s) > 0.2$)
- average number of translations per source word in the sentence (as given by IBM 1 table thresholded so that $prob(t|s) > 0.01$) weighted by the inverse frequency of each word in the source corpus

On the other hand, there has to be a mapping from this tuple to a y value, to represent the desired output.

This is done by using one of the metrics described in §2.4.2. For example a suitable y value could be the HTER of the $[trg, pe]$ pair.

Given these mappings, applying *online* learning to the QE task is straightforward:

In consecutive rounds $t = 1, 2, 3...$:

1. the model receives a vector $\mathbf{x}_t$ of the features extracted from the $[src, trg]$ pair and returns a prediction $\hat{y}$.
2. then it receives the true value y which responds to a metric of quality
3. and, using this feedback, the model is updated.

3.5.2 CAT-tool Framework

In the real-life scenario of a translator using a CAT-tool, in each round i of the *online* learning process:

1. the QE Server receives a $[src_i, trg_i]$ pair, as produced by the SMT system.
2. extracts the features $\mathbf{x}_i$ and produces a prediction $\hat{y}_i$
3. the post-editor corrects the sentence or writes it from scratch (hopefully being aided in his decision by the prediction of the QE Server)
4. the QE Server receives the post-edited sentence $[pe_i]$ and computes the true value y_i
5. the QE Server updates the model by training with the $(\mathbf{x}_i, y_i)$ tuple.

Chapter 4

An open-source infrastructure for Adaptive Quality Estimation

4.1 Introduction

The software package that we created includes all necessary components for a server that performs *Adaptive* Quality Estimation, with input that is given in an *online* fashion. It also includes other components for setting up a client for this server, which is needed for all the experiments.

The QEServer package supports different configurations for choosing the learning algorithms, their parameters, and other properties, such as storing the weights of the regressors, or for performing other actions, such as cross-validation after a certain amount of training points.

The QEClient package is only used to simulate the scenario of a post-editor using a CAT-tool, providing the QE Server with *[src,trg,pe]* tuples.

Both packages, along with a detailed manual, and demo data are available at `https://bitbucket.org/antonis/adaptiveqe`.

The rest of this chapter describes the basic user scenario and required system specifications. Then, the embedded and used libraries are roughly presented, along with the final architecture and other details of the implementation.

4.2 System Specifications

In this section the system specifications that will ensure usability of the current software are described.

According to the International Organization for Standardization (ISO), *usability* is defined as the capability of a system to provide effectiveness, efficiency, and subjective satisfaction to its users. As sub-characteristics of usability the following can be considered: learnability, understandability and operability.

4.2.1 Functional Requirements

Having in mind that the software was developed as part of the MateCat project, the most important requirements were the following:

- Since the QE Server would be embedded in the CAT-tool as a separate module, it should communicate with the other modules in the standardised way and provide output in the required format.
- The time needed for giving a prediction when a *[src,trg]* pair is received, and the time needed for updating the model when the *post-edited* sentence is received should be reasonably small. Otherwise, the module wouldn't be useful in a real-life scenario, since the overall goal of the CAT-tool is to increase the productivity of the translators.
- Since the MateCat project is still under development and research on QE is still going on, the QE Server modules should be designed and implemented in a modular way, so as to enable future additions or modifications to the software.
- The package should enable the user to keep track of everything that takes place while the software is running.
- The package should enable the user to choose the parameters of its use (learning method, learning rate, etc)

However, since the QE Server is also going to be released publicly to be used as a collection of libraries for *online* Quality Estimation, the package should also:

- Provide all the components (that would otherwise be part of of the CAT-tool) needed for the QE Server to run as standalone.
- Provide a simple Client for the QE Server, which will simulate a simple interface for experiments on *online* QE.

4.2.2 Functional Specifications

Each functional requirement is matched with functional specifications, which are the guidelines through which the project must be designed and implemented. This mapping is presented in table 4.1.

Requirements	**Specifications**
Communicating in a standardised way	Input and output are given by reading/writing from/to the *standard input/output*, as do the other MateCat components.
Input/Output in standardised format	Require input/output to be in the standardised format. Develop parsers for input and standard *print* methods for output.
Time Efficiency	Use C++ (considered quite fast) and optimize the code wherever possible.
Modularity	Use the *adapter* mode for all modules that are subject to change (see figure 4.2).
Have full visibility during the execution of the project	Use a log file to write all important steps taken by the QE Server and enable for various levels of verbosity.
Enable parameters choice	Use a 'properties' file with all parameters and their choices which is used for instantiating the parameters of the QE Server, during initialization.

TABLE 4.1: Functional Requirements and Specifications

4.2.3 Non functional Requirements

The non functional requirements, related to the nature of the project, include:

- Extensive documentation of all the components used and created, in order to be included in the documentation of the MateCat project.
- A User Manual that will guide a potential user into installing, configuring and running the QE Server outside a CAT-tool.

4.2.4 Development Platform

The software was developed in two phases. The first was the development in FBK, during which the operating system was Red Hat Enterprise Linux 6. The second phase was conducted in NTUA and the operating system was Ubuntu 12.04.

The programming language chosen for the implementation of the QE Server is C++, since it combines high-level object oriented features with the speed of a low-level programming language. In addition, several of the libraries implementing online learning methods are written in C++ and therefore would be easier and more efficient to embed in a C++ project rather than a project implemented in another programming language.

Given the complexity and the volume of the project and the estimated size of the code needed to implement it, the use of an established platform for large-scale C++ project development is vital. In both phases, the platform used is *NetBeans IDE 7.3.1.* (with the appropriate plugin to support C/C++ projects).

One advantage of *NetBeans* over other IDEs like eg. *Eclipse*, is that it supports virtual folders, so the organisation of the source code is quite easy, without need for complicated Makefiles.

The standalone version of the project, in addition, needs components that also require the Java and Perl programming languages.

Java is needed for the QuEst module. QuEst[1] is a tool for translation quality estimation Specia et al. (2013) and is running in parallel with the main QE system. The tool, which implements a large number of features proposed by participants to the WMT QE shared tasks, has been modified to process one sentence at a time, as requested for integration in the CAT framework and it is included as an executable .jar file. The "online" version of QuEst can also be downloaded from `http://www.quest.dcs.shef.ac.uk/`.

Perl, on the other hand, is needed for the Faucet code (myFaucet.pl) which is responsible for the connection between the language models and the QuEST module.

4.2.5 UML diagrams

The standard use case scenario is the following:

1. The CAT-tool provides the QE Server with a *[src,trg]* pair.
2. The QE Server responds with a prediction for this pair.
3. The translator post-edits the sentence and commits it, so that the CAT-tool can provide the QE Server with the *[pe]* of the tuple.
4. The QE Server updates the model and waits for the next instance.

The sequential diagram, showing the internal steps followed by the QE Server for this standard use case scenario, is presented in figure 4.1.

Figure 4.2 shows a general class diagram for the 'Adapter' pattern that is used for creating compatible interfaces for both the online learning libraries and the TER library. This programming pattern enables the main class to always use the same interface when communicating with any library that has certain functionalities.

[1] `http://www.quest.dcs.shef.ac.uk/`

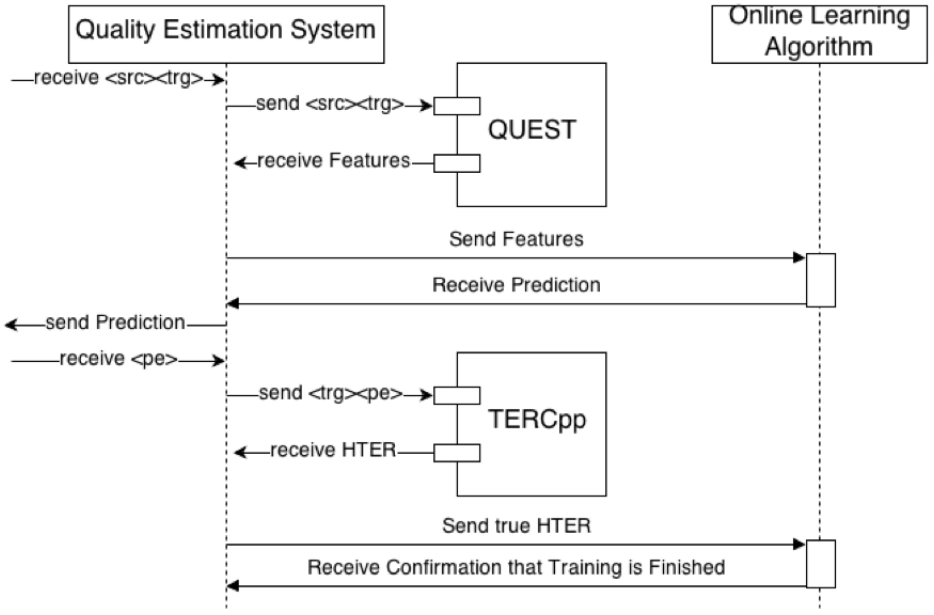

FIGURE 4.1: Sequence Diagram for the standard use case scenario

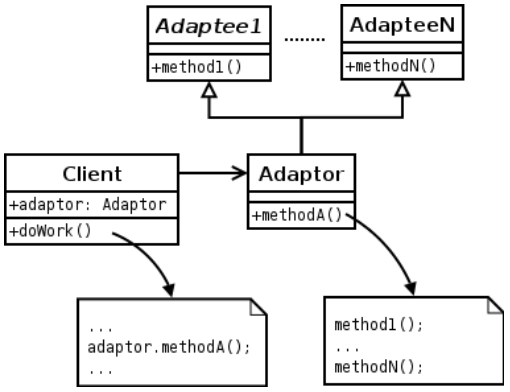

FIGURE 4.2: General Class Diagram for the Adapter pattern.

4.3 Libraries

This section presents the external libraries that were embedded in the code. The *TerCpp* library is responsible for calculating the HTER between the suggestion and the posti-edited sentence. The rest of the libraries implement or support the implementation of adaptive learning methods.

4.3.1 Online SVR Library

The Online SVR library was created by Parrella (2007) as part of his MSc thesis. (http://onlinesvr.altervista.org/)

It implements an online version of SGD-SVM, enabling selection for various kernels: LINEAR, POLYNOMIAL, RBF, and others. It also enables for parameters selection (ϵ, C, kernel parameters)

It implements three classes:

- *OnlineSVR*: The main class, which incorporates:
  - the parameters and the *Get(), Set()* methods for their manipulation
  - methods for Kernel function
  - methods that implement training and predicting
  - methods for input/output from/to files
  - methods for defining informational messages, according to the set verbosity
- *Vector*: The class that defines a vector and implements methods for:
  - *Set(), Get()* for basic parameters (length, cell values)
  - Basic Vector operations (Sum, Product, Power, Divide, Subtract, Min, Max, Mean, Variance)
  - Sorting operations
  - I/O operations
- *Matrix*: The class that defines a matrix building upon the class *Vector*, and implements methods for:
  - *Set(), Get()* for basic parameters (rows and columns length, cell values)
  - *Add(), Remove()* rows or columns
  - Basic Vector operations (Sum, Product, Power, Divide, Subtract)
  - Basic initialization operations (with random numbers or zeroes)
  - Sorting operations
  - I/O operations

This library is also the basis for the main software. We use the *Vector* and *Matrix* classes also for storing the *training* set and the instances that are received.

4.3.2 sofiaml library

SofiaML implements various online learning algorithms, such as Passive-Aggressive, SGD-SVM, Pegasos, Margin Perceptron and others. It was developed by Sculley (2010) and is hosted by google (https://code.google.com/p/sofia-ml/).

It focuses mostly on the speed of the learning process and the initial release intended to aid researchers and practitioners who require fast methods for classification and ranking on large, sparse data sets.

It implements 7 classes:

- 5 of these classes (*SFDataSet, SFHashInline, SFHasWeightVector, SFSparseVector, SFWeightVector* are dedicated to data representation, in order to deal with data sparseness and also aid with maximizing speed performance
- class *SofiamlMethods*, which defines the learning methods and several sampling Gradient Descent methods through which they can be applied for different tasks (binary classification, ranking, optimizing ROC area). The defined methods are:
  - Pegasos SVM
  - Stochastic Gradient Descent (SGD) SVM
  - Passive-Aggressive Perceptron
  - Perceptron with Margins
  - ROMMA
  - Logistic Regression (with Pegasos Projection)
- class *Sofiaml*, which incorporates:
  - the parameters and the *Get(), Set()* methods for their manipulation
  - methods for input/output from/to files

Since the original SofiaML distribution had only implemented online learning algorithms for the Classification problem, we had to modify the code in order to enable learning for the regression problem.

This modification only required changing the functions that calculate the $\epsilon-$insensitive loss function and the function for updating the weights of the model.

The only currently supported (and modified for the regression task) algorithm from SofiaML is the Passive-Aggressive algorithm.

4.3.3 newmat library

This C++ library (Davies, 2000) defines various classes, which enable manipulation for a variety of types of matrices using standard matrix operations.

Among the various classes, there are:

- Matrix Types:

  - Matrix (rectangular matrix)
  - UpperTriangularMatrix
  - LowerTriangularMatrix
  - DiagonalMatrix
  - SymmetricMatrix
  - BandMatrix
  - UpperBandMatrix
  - LowerBandMatrix
  - SymmetricBandMatrix
  - IdentityMatrix
  - RowVector
  - ColumnVector

- Matrix operations: product, sum, subract, Kronecker product, Schur product, concatenation, inverse, transpose, conversion between types, submatrix, determinant, Cholesky decomposition, QR triangularisation, singular value decomposition, eigenvalues of a symmetric matrix, sorting, fast Fourier and trig. transforms and printing.

4.3.4 OnlineGP library

This library was created by Grollman (2010), for his PhD Thesis in EPFL (http://lasa.epfl.ch/ dang/code.shtml). It implements the Sparse Online Gaussian Process learning algorithm. For representing vectors, matrices and their operations it uses the classes and methods defined by the newmat library.

Its classes are:

- *SOGPParams*: The class that defines the basic parameters (*capacity*, $s20$ and *kernel_type*)
- *SOGPKernel*: The class that defines the kernel used. It extends to:
  - *POLKernel*, which implements a polynomial kernel
  - *RBFKernel*, which implements an RBF kernel
- *SOGP*: the main class, which implements methods for:
  - *get(), set()* methods for basic parameters
  - learning operations (*add(), predict()*)
  - I/O operations from/to files

4.3.5 TerCpp library

Created by Cristophe Servan, this library implements Snover's algorithm for calculating the TER metric (http://sourceforge.net/p/tercpp/wiki/Home/).

It implements various libraries, for input/output from .txt and .xml documents and also for computing the alignments and shifts between the documents, which are needed for the TER calculation.

4.4 Final Architecture

4.4.1 Flowchart for *main.cpp*

The source file 'main.cpp' is the main function for this project. A - basic - flowchart is the following:

1. Connect to QuEST
2. Initialize the TER and Learning Adaptors *(more on the relevant section)*
3. Initialize the configuration, by parsing the 'PROPERTIES.CONFIG' file
4. Establish the connection with client (the server will wait until a client connects)
5. While the client is connencted:
   (a) Receive input
   (b) Parse the input (break it into separate queries) and store the queries in a queue.
   (c) While the queue is NOT empty:
      i. Take the head of the queue (query from client)
      ii. Parse it
      iii. Execute the appropriate action (give prediction or train or do nothing if the input is not structured correctly)
      iv. Respond appropriately to the client (send prediction or send confirmation that training is finished or send error message)
6. shut down the server if the client sends the appropriate message ('ENDEND') before disconnecting

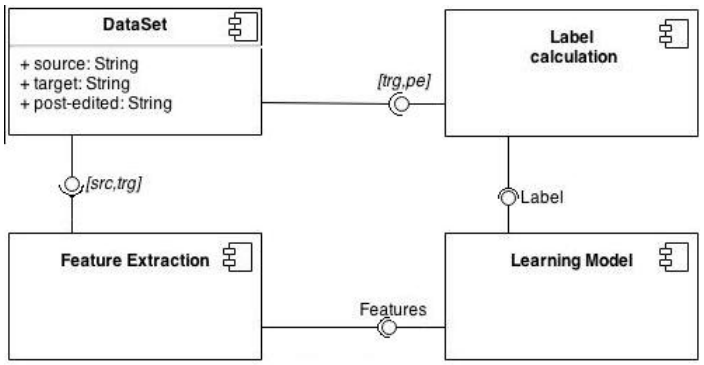

FIGURE 4.3: Component Diagram for the QE Server

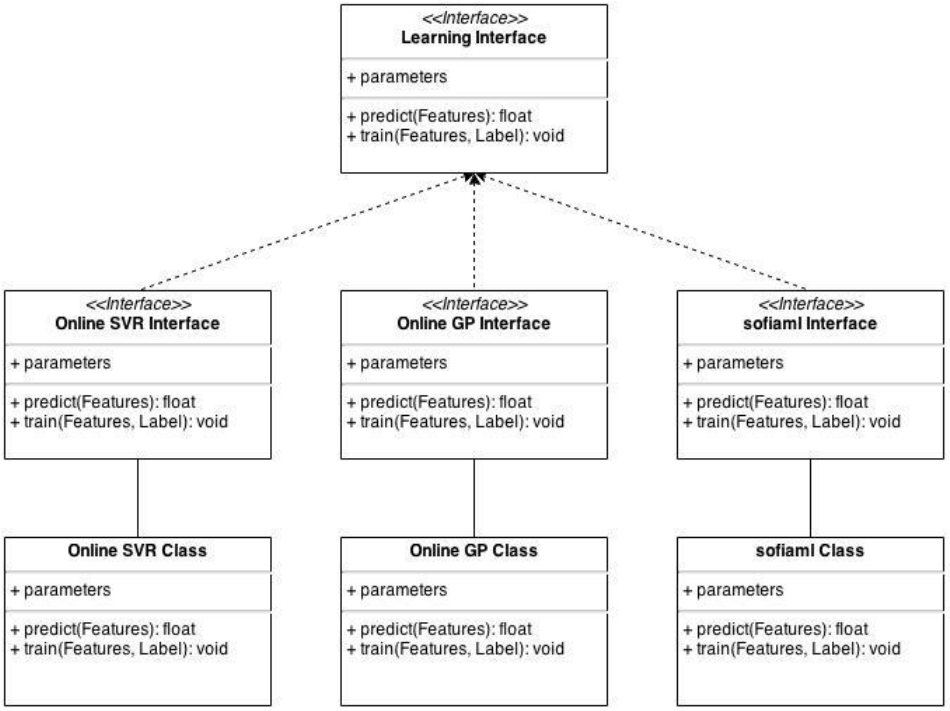

FIGURE 4.4: Class Diagram for the Learning Model Component.

4.4.2 QE Server Classes

In this section the main objects/classes of the QE Server are described. The general UML component diagram is presented in figure 4.3.

The rough UML class diagrams for the Learning Method component is presented in figure 4.4. The component for the calculation of the label is structured in the same way (using an public and a virtual interface, as instructed by the Adapter pattern).

4.4.2.1 TrainInstance

This is the class that represents the segments of the document. Its specifiers, which describe a whole training Instance object, are:

- **id**: the segment ID, which must be unique (provided by the parsed input)
- **src**: the source sentence
- **trg**: the target sentence
- **FeatureVals**: a hashmap with the values of the features (provided by QuEST) of the *[src,trg]* pair
- **hterPrediction**: the prediction for the HTER value, provided by the learning model
- **PostEditorID**: the ID of the post-editor. It can be the ID of the client.
- **pe**: the post-edited sentence
- **HTER**: the true value of the HTER (provided by TERCpp)

4.4.2.2 Features and Labels

The Features and the Labels of the segments, are not only stored in the TrainInstance object.

The main class also creates a Matrix '*FeaturesSet*' (for the Features) and a Vector '*Labels*' (for the Labels). This matrices are storing incrementally the features and labels needed for re-training the model.

For Each new training instance, its features are added to this *FeaturesSet* matrix and its Label is added to the *Labels* vector. These matrices are, then, the parameters of the 'Train' function of the Learning Interface.

4.4.2.3 Online Learning Libraries Classes

Each online learning library embedded in the project is represented as a class, with its own specifiers and functions. These classes are not changed from their original distribution. As described in section 4.3 the online learning libraries embedded are:

1. OnlineSVR
2. OnlineGP

3. SofiaML

The functions of the learning libraries are not used directly from the main, but there is an intermediate level between the main and libraries. This level is the Learning Adapter, which provides an adaptor to all the supported learning algorithms.

That way, the main code always calls the same function, eg. 'Predict', and the adaptor selects the correct function according to the selected learning algorithm.

4.4.2.4 Learning Interface

This is the class of the interface of the adaptors for the online learning libraries. Its specifiers are virtual functions. These virtual functions are the ones used in the main programme to address the learning libraries.

The virtual functions are mapped to the 'original' ones for each learning algorithm and whenever called, they point to the correct function to be used.

The mapping from the virtual to the real functions is provided independently for each library, through their specific interfaces:

- **OnlineSVRInterface**: Includes the mapping from the virtual functions to the functions of the OnlineSVR class.
- **OnlineGPInterface**: Includes the mapping from the virtual functions to the functions of the OnlineGP class.
- **SofiaInterface**: Includes several interfaces, one for each learning algorithm supported by SofiaML.

  Each interface includes the mapping from the virtual functions to the functions of the SofiaML-[Algorithm] class.

  Note that the *sofiaml* interfaces also implement different 'Predict' and 'Train' functions (than the ones used for *OnlineSVR*) because they also implement and incorporate the **normalisation of the data** (which is not needed by *OnlineSVR* since it was already implemented in its class, but necessary for *sofiaml*).

4.4.2.5 TER and TERInterface

This is the class that embeds the *TERCpp* code into the project. It receives a pair of *[trg,pe]* sentences and returns the HTER score, which shows their 'post-edit difference'.

Again, there is an intermediate level between TERCpp and the main programme, implemented with the "Adapter" pattern.

The TERAdaptor specifies only one virtual function ('execute') which receives the sentence pair and returns the HTER score.

The TERInterface class implements the interface of the TERAdaptor, through which one can set the configuration of TERCpp (eg. choose whether to tokenize the text or not, choose whether to take punctuation into account, etc).

Chapter 5

Experiments with English-Spanish

5.1 Introduction

In this section we present two sets of experiments, with an artificially created English-Spanish dataset.

First, we experiment with different training set sizes and algorithms, in order to investigate how their performance is affected by the training set size.

Second, we present experiments with completely different label distributions, aiming to show that *batch* methods are not the best choice for such cases, when compared to *adaptive* methods. As discussed in chapter 3, differences in the label distribution between the *training* and *test* set result in a drop of performance, which can hopefully be addressed by using *adaptive* methods.

Meanwhile, we also compare the various *online* algorithms among themselves, in order to determine their differences and whether one is outstandingly better than the others.

The results indeed confirm that in datasets with homogeneous label distribution the *adaptive* algorithms are as good as the *batch* ones regardless of the *training* set size, but they are significantly better in cases where the label distribution across *training* and *test* set is different.

The final section of the chapter provides a summarization of the observed results and tries to explain them, viewed from various points of view.

5.2 Experimental Framework

The following set of experiments is carried out in controlled conditions. This means that the dataset is artificially obtained, but also that the *training* and *test* sets are chosen beforehand in order to certify certain conditions.

Using English-Spanish tuples of *[src,trg,pe]* sentences, for all the algorithms we train and test according to the process described in §3.

5.3 Dataset

The dataset used for experiments 1 and 2 is an English-Spanish corpus, which was created for the Shared QE task of the WMT12 workshop (1832 for *training* and 422 for *test*).

All sentences are from the *news* domain. *Target* sentences are produced by a phrase-based SMT system (Moses) trained on Europarl and News Commentaries corpora as provided by WMT.

The HTER labels for our regression task are calculated from the post-edited version of the *target* sentences provided in the dataset.

5.4 Homogeneous label distribution

The motivations for experiments with training and test data featuring homogeneous label distributions are twofold.

Firstly, since in this artificial scenario adaptation capabilities are not required to the QE component, *batch* methods operate in the ideal conditions. This makes possible to obtain from them the best possible performance to compare with.

Secondly, this scenario provides the fairest conditions for such comparison since, in principle, *online* algorithms are not favoured by the possibility to learn from the diversity of the test instances.

5.4.1 Experimental Setup

In order to avoid biases in the label distribution, the WMT12 training and test data have been merged, shuffled, and eventually separated, in order to generate three *training* sets of different size (200, 600, and 1500 instances), and one *test* set with 754 instances.

For each algorithm, an *adaptive* and an *empty* model are compared against a Support Vector Regressor trained in batch mode. The *adaptive* model builds upon a model created[1] from the training data and also exploits the test instances to stepwise refine its predictions. The *empty* model learns from scratch from the test set, not having been trained on any instances before. The *empty* model simulates the worst, but also realistic, conditions, were no training data are available for the creation of a model.

The notation that is used for the next experiment is ALG_i, where ALG refers to the model obtained using a given learning algorithm and i refers to the *training* sets. The parameter i can take the values 200, 600, 1500, referring to the *training* set with the equivalent size, whereas it is always the same *test* set that is used throughout this experiment.

In Table 5.1 is presented the distribution of the labels across the *training* and the *test* sets. It is clear that all sets reflect the same distribution, since the differences that are observed are not significant.

This can be further supported by using a distribution similarity metric. A suitable such metric is the Hellinger distance (Hellinger, 1909), which, for two distributions[2] $P \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2^2)$ can be computed as:

$$H^2(P,Q) = 1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2}}\, e^{-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}}.$$

The closer to 0 the Hellinger distance is, the more similar the two distributions.

Training Labels			Test Labels		Distribution
Training Set	Avg. HTER	St. Deviation	Avg. HTER	St. Deviation	Distance
200	32.71	14.99	32.32	17.32	0.072
600	33.64	16.72			0.033
1500	33.54	18.56			0.042

TABLE 5.1: Label Distribution across *training* and *test* sets for Experiment 1.

For each algorithm, the training sets are used to learn the QE models, optimizing parameters through grid search in 10-fold cross-validation. For each algorithm, the parameters that are optimized are:

- SVR-OSVR: C, ϵ and the kernel type
- PA: C (aggressiveness) and ϵ (tolerance)
- OGP: the capacity of the $\mathcal{BV}$ vector.

[1] Training is also performed in an online fashion.
[2] We show in section §6.4 that the labels' distributions are indeed Gaussian.

Evaluation is carried out by measuring the adaptability of *batch*, *adaptive*, and *empty* models in terms of global MAE scores on the test set.

It is important to note that, in order for the results to be completely comparable, the parameters used in the experiments with the *empty* models are the ones that resulted from the optimization for the *adaptive* models.

Computing the statistical significance of the results will also enable us to further understand the results. The null hypothesis in all the next experiments is that the *batch* and *online* methods perform differently, by providing different predictions. With approximate randomization (Yeh, 2000), we calculate the probability that this null hypothesis can be rejected. If this probability p is lower than 0.005, then the null hypothesis stands and the differences of the performance are significant. In the inverse case, when $p > 0.005$, the differences in the performance of the algorithms are not statistically significant.

The predictions of the best *batch* model are compared to the predictions of the best *online* model. If the statistical significance test gives that $p < 0.005$, then the difference in the performances of the best *online* model and the best *batch* model is statistically significant. Otherwise ($p > 0.005$), the results of the *online* and *batch* models do not differ significantly and are marked with an asterisk (*)[3].

5.4.2 Experiment 1: Varying the size of the Training Set

The results of Experiment 1 are reported in Table 5.2. The *batch*, *adaptive* and *empty* models are compared for the three *training* sets. A similar behaviour is visible for all the algorithms.

With the same amount of training data, performance differences between batch and adaptive models are rather small. Even in the worst case, the performance of the worst *online* model ($OSVR-RBF$) is only one MAE point worse that the performance of the best *batch* model ($SVR-RBF$); this is observed in the case were we have 600 *training* instances.

This demonstrates that, as expected, the online algorithms do not take further advantage from test data with a label distribution similar to the training set and do not improve further than the *batch* models. On the other hand, they do manage to achieve performance comparable to the performance of the *batch* models. In fact, the difference between the results of the best *batch* and the best *online* model is not statistically significant. Thus, we can conclude that, in this experiment, the *online* models perform almost as good as the *batch* models.

[3]Statistical significance is always calculated in the same way and the same notation is used for the rest of the thesis

Algorithm	Kernel	MAE ($i = 200$)	MAE ($i = 600$)	MAE ($i = 1500$)
Batch				
SVR_i	Linear	13.5	13.0	12.8
	RBF	**13.2***	**12.7***	**12.7***
Adaptive				
$OSVR_i$	Linear	**13.2***	12.9	12.8
	RBF	13.6	13.7	13.5
PA_i	-	14.0	13.4	13.3
OGP_i	RBF	**13.2***	12.9	12.8
Empty				
$OSVR_0$	Linear	13.5		
	RBF	13.7		
PA_0		14.4		
OGP_0	RBF	13.3		

TABLE 5.2: MAE of batch, adaptive and empty models on data with homogeneous label distributions.

At the same time, however, these results bring some interesting indications about the behaviour of the different online algorithms.

First, the good results achieved by the empty models (less than two MAE points separate the worse model, PA_0^{754}, from the best ones built on larger *training* sets) suggest their high potential when training data are not available.

Second, the results show higher MAE variations but slightly worse results for PA than for OSVR and OGP. While for PA the MAE of empty and adaptive models ranges from 13.3 to 14.4, the results for OSVR (with Linear kernel) range from 12.8 to 13.5 and for OGP (RBF kernel) range from 12.8 to 13.2. This difference in performance suggests a slightly lower capability of PA, compared to OSVR and OGP, to learn from new instances.

5.5 Disjoint label distributions

In order to explore the full potential of adaptive models in a controlled scenario, it is interesting to investigate the situation where the *training* and *test* sets reflect completely different label distributions.

On the one hand, this setting provides the worst conditions for *batch* learning. Since the *training* and *test* sets are completely different, and the *batch* model can only produce predictions based on data of the *training* set, its performance on the *test* set is expected

to drop, compared to the excellent results of the previous scenario. On the other hand, this is a more favourable situation for *online* methods, since they have the advantage to learn from user feedback also in the *test* set, gradually adapting their predictions to the new label distribution.

The following experiment aims to verify these intuitions.

5.5.1 Experimental Setup

In order to obtain maximally different label distributions, the WMT12 training and test data have been merged, sorted based on the HTER labels and eventually separated to generate two datasets.

The first one (*Top*) contains the top 600 instances, where the high HTER labels (ranging from 0.43 to 1) indicate a poor translation quality. The second one (*Bottom*) contains the 600 instances with the lowest HTER labels (from 0 to 0.21) indicating a good translation quality. The two datasets are alternatively used for training and test.

As in §5.4.1, for each algorithm, an *adaptive* and an *empty* model are compared against a Support Vector Regressor trained in batch mode.

The notation that is used for the next experiment is ALG^{test}_{train}, where ALG refers to the model obtained using a given learning algorithm and *train* and *test* refer to the *training* and *test* sets. The parameters *train* and *test* take the values *Top* and *Bottom* alternating referring to the *training* sets mentioned in the previous paragraph.

The distribution of the labels across the *training* and the *test* sets is presented in Table 5.3. It is clear that the two sets reflect disjointed label distributions. Another factor that could affect the performance of the learning methods is the different standard deviation (and, naturally, different range) that the two sets cover, with the *bottom* set having a range of [0, 21.05] HTER points, whereas the *top* set features instances within the range of [43.48, 1] HTER points, almost three times the range of the *bottom* set.

The Hellinger distance between the *Top* and the *Bottom* distribution is

$$H(Top, Bottom) = 0.95.$$

Note that the highest possible Hellinger distance between two distributions is 1, meaning that the *Top* and *Bottom* distributions are quite different.

Set	Average HTER	HTER St. Deviation
Top	56.27	12.59
Bottom	12.35	6.43

TABLE 5.3: Label Distribution for *Top* and *Bottom* sets for Experiment 2.

Again, as in §5.4.1, for each algorithm, the training sets are used to learn the QE models, optimizing parameters through grid search in 10-fold cross-validation.

Evaluation is carried out by measuring the adaptability of *batch*, *adaptive*, and *empty* models in terms of global MAE scores on the test set.

5.5.2 Experiment 2: Testing on different label distribution

The results reported in Table 5.4 confirm the expectations about the potential of online models. All algorithms, OSVR, OGP and and PA achieve significantly better performance than SVR, with MAE reductions of about 15 points on the *Top* test set and at least 8 points on the *Bottom* one.

Test on *Top*		
Algorithm	Kernel	MAE
Batch		
SVR^{Top}_{Bottom}	Linear	43.7
	RBF	43.2
Adaptive		
$OSVR^{Top}_{Bottom}$	Linear	28.7
	RBF	31.1
PA^{Top}_{Bottom}	-	28.2
OGP^{Top}_{Bottom}	RBF	**27.2**

Test on *Bottom*		
Algorithm	Kernel	MAE
Batch		
SVR^{Bottom}_{Top}	Linear	39.3
	RBF	40.7
Adaptive		
$OSVR^{Bottom}_{Top}$	Linear	**27.0**
	RBF	29.5
PA^{Bottom}_{Top}	-	31.0
OGP^{Bottom}_{Top}	RBF	28.3

TABLE 5.4: MAE of *batch* and *adaptive* models on data with disjointed label distributions.

These results confirm that *batch* learning methods cannot cope with differences between the *training* and *test* sets, whereas *online* approaches manage to perform better.

In contrast with the results of previous experiments, when dealing with disjointed label distributions we do not observe systematic differences between the *online* algorithms, with the best results achieved by OGP and OSVR for the two datasets.

Algorithm	Kernel	MAE on *Top*	MAE on *Bottom*
Empty			
$OSVR_0$	Linear	8.42	5.67
	RBF	8.55	5.37
PA_0	-	**8.37**	5.30
OGP_0	RBF	8.83	**5.22**

TABLE 5.5: MAE of *empty* models on data with disjointed label distributions.

No significant differences between algorithms are also observed in the results achieved by the *empty* models, reported in Table 5.5, which in all cases are very good and similar

N^o of instances	$OSVR_{LINEAR}$	$OSVR_{RBF}$	PA	OGP_{RBF}
Training Time (ms)				
954	44.10	34.69	7.56	10.69
1354	120.55	227.38	8.87	11.47
2254	277.71	318.62	11.79	13.75
Predicting Time (ms)				
954	0.364	1.254	0.996	38.35
1354	0.510	1.513	1.925	38.41
2254	0.759	2.053	4.517	37.85

TABLE 5.6: Average *training* and *predicting* time for the *adaptive* algorithms.

to each other. In these cases the MAE on both test sets is always below 9 points. This excellent performance can be explained by the artificial construction of the two sets.

Due to the small range and small standard deviation of the labels in each set, an *empty* model can quickly converge and eventually overfit the data. Note that, the smaller the range (as in the *Bottom* set), the better the performance. The three times larger range of the *Top* set, results in 3 MAE points worse performance, compared to the *Bottom* set.

In this setting, however, the good results of the empty models are not particularly informative since the artificial distortions in the data labels produce an unrealistic scenario where training data are more harmful than useful (thus penalizing *batch* and *adaptive* models).

5.6 Time performance of the algorithms

This section analyses the efficiency of the adaptive algorithms in terms of the time needed to provide a prediction and to update the model.

For the three modes of experiment 1 (§5.4.2, training with 200,600 and 1500 instances) the average training and prediction time for the algorithms is computed and reported in table 5.6. Note that the final sum of the instances is, 954, 1354 and 2254 for each case respectively.

It is obvious that all algorithms and kernels need more time to train as the instances increase. The prediction time for all algorithms is very low, ranging from less than 1ms to almost 40ms, a range more than acceptable for the CAT-tool.

However, *PA* needs significantly less time than the others for training. The times of *OGP* are also quite competent. By far, the worse times are the ones of *OSVR* (still acceptable for a CAT-tool, though); this can be attributed to the complexity of the *OSVR* algorithm and to the fact that it needs to explicitly use all previous instances in every iteration.

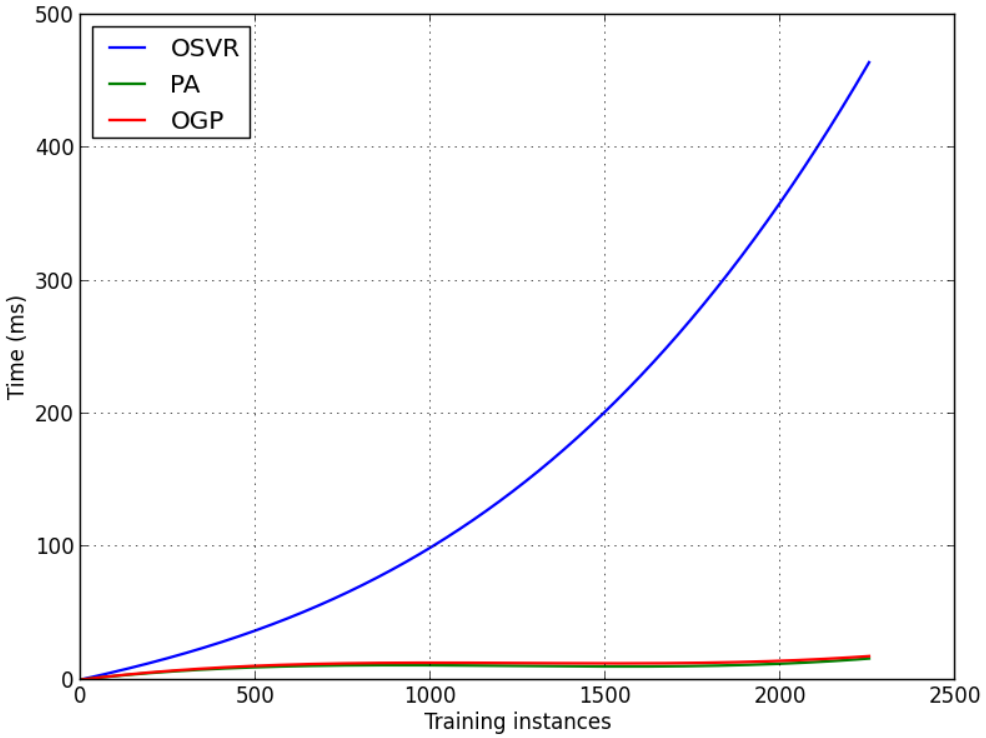

FIGURE 5.1: Updating Time according to training instances

The same results are also evident in the trend plots shown in figure 5.1, which show how the update time for the model of each algorithm is affected by the number of the previous *training* instances.

5.6.1 Computational Complexity of the Algorithms

Another aspect that would explain the time performance of the online methods is their computational complexity.

Given a number of seen samples n and a number of features f for each sample, we calculate the computational complexity of updating a trained model with a new instance for each of the methods we use:

- The complexity for training standard (not online) Support Vector Machines is $\mathcal{O}(n^2 f)$(Bottou and Lin, 2007) but it depends on the implementation and the various optimizations suggested.
- The complexity for updating a trained model with *OSVR* is in the worst case $\mathcal{O}(n^3 f)$ and in the average case it is $\mathcal{O}(n^2 f)$ (Parrella, 2007).
- The complexity for updating a trained model with the Passive-Aggressive algorithm is $\mathcal{O}(f)$ as the previous instances are not taken in account.

- For an Online Gaussian Process method with bounded $\mathcal{BV}$ vector with maximum capacity d, the complexity of a single update of the model is bounded by $\mathcal{O}(nd^2f)$ (Csató and Opper, 2002), and on run-time it is $\Theta(n\hat{d}^2f)$ where $\hat{d}$ is the actual number of vectors in the $\mathcal{BV}$ vector.

The fact that *OSVR*'s complexity is quadratic with respect to the previously seen instances justifies the time plot seen in figure 5.1. The similarity in *PA*'s and *OGP*'s time performance can be justified by the following facts:

1. For both *PA* and *OGP* there is a preprocessing step taking place, where the features of the new instance are normalised according to the previously seen ones, a calculation with $\Theta(nf)$, for n previously seen instances with f features. This adds up to the the actual algorithm's complexity, only affecting *PA*'s complexity and making it also bounded by $\mathcal{O}(nf)$.
2. In our experiments, the actual value of d for the $\mathcal{BV}$ vector of *OGP* does not exceed 100 or 200, according to the configuration, thus it does not affect significantly the time performance of *OGP*.

5.7 Summary of the English-Spanish results

The results of the previous sections allow us to draw useful conclusions. Firstly, regarding the performance of *online* methods against *batch* methods:

- When dealing with homogeneous label distributions, *batch* methods perform slightly better than *online* methods, but not significantly better. When the *training* data are relatively fewer than the *test* data, *online* and *batch* perform equally well.
- When dealing with disjoint label distributions, *online* methods perform significantly better than the *batch* methods, adapting to the *test* set at a satisfactory level.
- *Empty adaptive* models, also produce competent results, showing that using *online* models when no training data are available, can be useful. Especially when the distribution of the HTER labels is narrow (with very small standard deviation), as in the cases of *Top* and *Bottom*, *empty* models seem to be the best choice.

In general, we could claim that, *adaptive* methods are not worse than *batch* methods when dealing with similar data distributions and are not only much better when dealing with different distributions of *training* and *test* set labels, but also quite competent when no *training* data are available. Thus, this advantage of the *online* methods makes them a good option for use in a CAT-tool.

In addition we can draw conclusions regarding the performance of each *online* learning method:

- *Online SVR* and *Online GP* show stable and coherent performance across the experiments and are the *online* methods that perform as good as the *batch* method in Experiment 1.
- *PA* shows slightly worse performance that the other two in most cases (the only exception is testing on the *Top* set in experiment 2), but shows good performance when the HTER label distribution is narrow.

As regards *OSVR*, we also compare the significance of the difference of the results, when using RBF and when using Linear kernel. In almost all cases these differences turn out not to be significant.

To conclude, probably *OSVR* or *OGP* would be a better choice than *PA* as default methods for a potential CAT-tool. Should one also require time efficiency, then the best solution is *OGP*, since its update or prediction time is stable and does not heavily depend on the number of instances that are being processed.

Chapter 6

Experiments with English-Italian

6.1 Introduction

In this section we present a series of experiments, with an English-Italian dataset, which reflects much more accurately a real translation job than the dataset of the previous experiments of §5.

Section 6.3 describes the dataset, which provides instances for different translators over the same document, for documents coming from two domains, IT and Legal.

A number of evaluation settings can be obtained by considering data from different post-editors. These range from simpler situations where training and test data come from post-editors with similar post-editing behaviour (either both conservative or both aggressive), to harder situations where training and test data come from post-editors with different behaviour (one conservative and one aggressive).

We take advantage of the characteristics of the dataset to experiment on the performance of the adaptive methods:

- within particular translation domains,
- when the *training* and *test* sets are created by different translators, but are both in the same translation domain,
- when the *training* and *test* sets fall within different domains and are created by different translators.

Section §6.4 describes an effort to model post-editors behaviour according to the data that they have provided, by suggesting several metrics. This section motivates the use of HTER as a metric of the translators' post-editing behaviour in the experiments' section.

The following sections describe the experiments in detail. In each experiment, the results are constituted by calculating global Mean Average Error scores and by presenting point-wise error plots.

The MAE aids in comparing the *adaptability* of each algorithm to heterogeneous test sets, obtained from different users or domains. The introduction of the point-wise plots, instead, aims at a more fine-grained analysis of the *sensitivity* of the *online* models to such differences.

For each experiment, in order to enable better understanding of its results, we provide a short summary of the information presented in the tables and the figures in the form of answers to the following questions:

1. Do *online* methods perform better than *batch* methods?
2. Which *adaptive* method performs better?
3. Do *empty* methods provide competent results, compared to the *adaptive* version?

The "Results Discussion" sections provide an overview of the results, along with explanations for the variations on the performance of the *adaptive* methods. The results show that *online* methods for QE can be a way to handle changes in the user or the translation domain, as is expected in the user scenaria of a CAT-tool.

6.2 Experiment Framework

The following set of experiments is carried out in more realistic conditions, compared to the experiments of §5. The datasets are obtained directly from translation jobs, involving different human translators working on texts from different domains.

All algorithms are trained and tested using English-Italian tuples of *[src,trg,pe]* sentences, according to the process described in §3.

6.3 Dataset

The dataset for the following experiments is constituted of two sub-datasets. The first one is a dataset of English-Italian tuples of *[src,trg,pe]* sentences, coming from the information technology (henceforth IT) domain. The second is another dataset of English-Italian tuples of *[src,trg,pe]* sentences, coming from the Legal (henceforth L) domain.

The source sentences were translated with a SMT system developed with the Moses toolkit (Koehn et al., 2007) trained on parallel data from the domain.

Post-editions were collected from four professional translators operating with the CAT tool in real working conditions. The four professional post-editors were faced with a document to translate and presented with MT suggestions for each sentence, that they could either accept or post-edit to reach publication quality. Note that the translators were not the same for the IT and the Legal document.

6.3.1 IT Domain Dataset

The dataset for the IT domain is taken from a software user manual, containing 280 sentences.

For the IT domain, training data (about 2M parallel sentences) were extracted from the OPUS corpus (Tiedemann, 2012), and from a proprietary translation memory built from real translation projects, provided by TRANSLATED (http://www.translated.net/en/), a commercial translation company and partner of the MateCat project.

The data for the label's distribution and the variation among different translators, computed in the whole IT document, are presented in Table 6.1.

Post-editor	Avg HTER	HTER St. Deviation
1	39.32	21.03
2	47.77	20.49
3	37.72	20.05
4	36.60	19.71

TABLE 6.1: Label Distribution across the IT document for each post-editor.

From table 6.1, differences among the various post-editor's pairs are visible. This different or similar behaviour of the translators is further discussed in §6.4

6.3.2 Legal Domain Dataset

The L document, extracted from a European Parliament resolution published on the EUR-Lex platform,[1] contains 164 sentences.

Training data for the legal domain (about 1.5M segments) come from the JRC-Acquis collection (Steinberger et al., 2006).

The data for the label's distribution and the variation among different translators, computed in the whole L document, are presented in Table 6.2.

Similar to the IT domain, the L data make possible to obtain different evaluation settings featuring different levels of complexity depending on the post-editors behaviour. In

[1] `http://eur-lex.europa.eu/`

Post-editor	Avg HTER	HTER St. Deviation
1	29.04	16.84
2	32.33	18.87
3	43.25	14.86
4	23.52	15.80

TABLE 6.2: Label Distribution across the Legal document for each post-editor.

table 6.2, differences among the various post-editor's pairs are notable. This different or similar behaviour of the translators is further discussed in §6.4.

6.4 Modelling Post-Editor Behaviour

This section attempts to deal with the problem of modelling the behaviour of human post-editors, through which identifying differences or similarities might be possible.

In order to confirm that there are indeed different behaviours, we compute the probability distribution of each posteditor's *HTER* labels, by estimating the kernel density function.

These *HTER* probability distributions are presented in figure 6.1. The difference, or similarity, of these distributions can be roughly identified even by the naked eye. To begin with, all post-editors' labels follow a gaussian distribution, a finding that is also confirmed for all post-editors, using the $\mathcal{X}^2-$Goodness-of-Fit method (Nikulin, 1973). However, certain post-editors' work results in gaussian distributions with 1 mode, other's results in gaussian distributions with two modes, or with right skewed gaussians.

The rest of the chapter suggests several metrics that could be used for measuring differences or similarities in the post-editing behaviour of different post-editors.

6.4.1 Metrics of similarity between post-editors

Through the following subsections only the results for the Legal document are presented as examples. The results of the IT document can be found in the Results section (§6.4.9).

All rankings tables show the pairs of post-editors in descending order of similarity. This means that, in the first row the more similar post-editors are presented, and in the last row the most different post-editors, according to each metric. The values of each metric for the pairs are presented in parenthesis.

In the rankings that are produced from each metric, the results of the previously mentioned metrics are also presented for easier comparison.

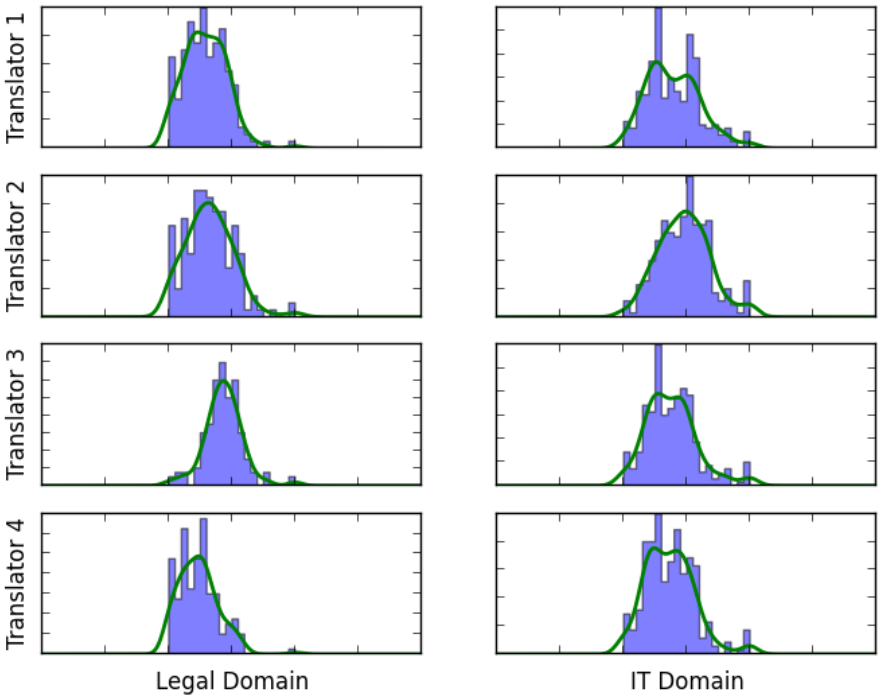

FIGURE 6.1: HTER Probability Distributions for all post-editors

6.4.2 Average HTER and St. Deviation

For each post-editor, the Average HTER and the Standard Deviation of the distribution of the labels that he assigns are calculated. The pairs of post-editors are ranked according to the difference of their average HTERs (the difference of the standard deviations are not really informative)

6.4.3 Vocabulary Size

This metric compares the vocabularies of the post-editors (the number of distinct words that they use). For each pair of post-editors, the size of the intersection of their vocabularies is calculated. The size is presented as the percentage of the vocabulary of each translator and the pairs are ranked according to the average percentage of vocabulary that they share in common. Results are presented in table 6.3.

6.4.4 n-grams

For each post-editor, we compute the 2-, 3-, 4- and 5-grams of the document that he has translated. For each pair of post-editors, we compute the percentage of n-grams that

the post-editors have in common. The ranking for each pair is produced by averaging over the two post-editors of the pair and the results are presented in table 6.3.

HTER diff	**Vocabulary**	**2-gram**	**3-gram**	**4-gram**	**5-gram**
1,2 (3.29)	1,4 (0.79)	1,4	1,4	1,4	1,4
1,4 (5.52)	1,3 (0.77)	1,2	1,2	1,2	1,2
2,4 (8.81)	3,4 (0.76)	2,4	2,4	2,4	2,4
2,3 (10.92)	1,2 (0.76)	1,3	1,3	1,3	1,3
1,3 (14.21)	2,4 (0.75)	3,4	3,4	3,4	3,4
3,4 (19.73)	2,3 (0.72)	2,3	2,3	2,3	2,3

TABLE 6.3: Ranking of all pairs for the Legal document according to n-grams metrics.

The results do not vary when using 2,3,4 or 5-grams, thus from here on they will be presented as "n-grams". The score will be computed by multiplying the scores for 2,3,4 and 5-grams, as in table 6.4.

HTER diff	**Vocabulary**	**n-grams**
1,2 (3.29)	1,4 (0.79)	1,4 (55.5)
1,4 (5.52)	1,3 (0.77)	1,2 (48.9)
2,4 (8.81)	3,4 (0.76)	2,4 (36.1)
2,3 (10.92)	1,2 (0.76)	1,3 (18.7)
1,3 (14.21)	2,4 (0.75)	3,4 (14.6)
3,4 (19.73)	2,3 (0.72)	2,3 (11.8)

TABLE 6.4: Ranking of all pairs for the Legal document according to n-gram metric.

6.4.5 Average Overlap

For every pair of post-editors, this metric computes the average length of the longest common sub-sequence, normalised by the average length of the sentences. The results are shown is table 6.5.

The results show that, either by using normalisation, or not, the rankings are the same, so henceforth only the normalised version will be presented.

HTER diff	**Vocabulary**	**n-grams**	**Norm. Overlap**
1,2 (3.29)	1,4 (0.79)	1,4 (55.5)	1,4 (33.1)
1,4 (5.52)	1,3 (0.77)	1,2 (48.9)	1,2 (9.4)
2,4 (8.81)	3,4 (0.76)	2,4 (36.1)	2,4 (9.2)
2,3 (10.92)	1,2 (0.76)	1,3 (18.7)	1,3 (6.9)
1,3 (14.21)	2,4 (0.75)	3,4 (14.6)	3,4 (6.6)
3,4 (19.73)	2,3 (0.72)	2,3 (11.8)	2,3 (5.9)

TABLE 6.5: Ranking of all pairs for the Legal document according to the overlap metric.

6.4.6 Distribution Difference

This metric quantifies the similarity of the distributions of the labels of two post-editors. We use two established metrics (Cha, 2007, Deza and Deza, 2009) for comparing distributions:

- Bhattacharyya distance (D_B)
- Hellinger distance (D_H)

The Bhattacharyya distance is defined by $D_B = -ln(BC(p,q))$, where $BC(p,q) = \Sigma_{x \in X} \sqrt{p(x)q(x)}$ is the Bhattacharyya coefficient. The Hellinger distance can also be defined by the BC as it stands that $D_H = \sqrt{1 - BC(p,q)}$.

For two gaussian distributions p, q, with means μ_p, μ_q and variances σ_p, σ_q (we assume that the HTER labels of the post-editors follow a gaussian distribution), the distances are defined as:

$$D_B(p,q) = \frac{1}{4} \ln \left(\frac{1}{4} (\frac{\sigma_p^2}{\sigma_q^2} + \frac{\sigma_q^2}{\sigma_p^2} + 2) \right) + \frac{1}{4} \left(\frac{(\mu_p - \mu_q)^2}{\sigma_p^2 + \sigma_q^2} \right)$$

$$D_H^2(P,Q) = 1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}} \, e^{-\frac{1}{4} \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}}$$

It stands $0 < D_B, D_H < 1$ but the triangle inequality is obeyed only by D_H. A result of $D_B = 0$ or $D_H = 0$ means that the distributions are the same, whereas a result of $D_B = 1$ or $D_H = 1$ means that the distributions are completely different. Of course, $D_{B/H}(p,q) = D_{B/H}(q,p)$.

The ranking of the pairs according to these metrics is shown in table 6.6

HTER diff	**Vocabulary Size**	**n-grams**	**Norm. Avg. Overlap**	**Distr. Diff. (Bhattach.)**	**Distr. Diff. (Hellinger)**
1,2 (3.29)	1,4 (0.79)	1,4 (55.5)	1,4 (33.1)	1,2 (0.007)	1,2 (0.007)
1,4 (5.52)	1,3 (0.77)	1,2 (48.9)	1,2 (9.4)	1,4 (0.015)	1,4 (0.015)
2,4 (8.81)	3,4 (0.76)	2,4 (36.1)	2,4 (9.2)	2,4 (0.04)	2,4 (0.039)
2,3 (10.92)	1,2 (0.76)	1,3 (18.7)	1,3 (6.9)	2,3 (0.066)	2,3 (0.066)
1,3 (14.21)	2,4 (0.75)	3,4 (14.6)	3,4 (6.6)	1,3 (0.104)	1,3 (0.099)
3,4 (19.73)	2,3 (0.72)	2,3 (11.8)	2,3 (5.9)	3,4 (0.208)	3,4 (0.188)

TABLE 6.6: Ranking of all pairs for the Legal document according to the distribution distance metrics.

In the next tables we will only use the *Hellinger* distance, since the results do not vary between these two metrics of distribution distance.

6.4.7 Instance-wise difference

The previous metrics are calculated using the means and variances of the distributions. However, two post-editors might have a "similar" distribution of labels, but when compared instance by instance they might prove to be quite different.

Since the post-editors work on the same document, we can compare their HTER labels one by one, and find, for each instance, the pair that has the most similar and the most different pair of labels. The metric counts, for each pair, the number of instances for which the pair shows the highest, or lowest HTER difference. The pairs are then ranked accordingly, in table 6.7.

Note: This metric is only applicable in the in-domain scenario, where the instances are comparable.

HTER diff	**n-grams**	**Norm. Avg. Overlap**	**Distr. Diff. (Bhattach.)**	**Distr. Diff. (Hellinger)**	**#times Different**	**#times Similar**
1,2 (3.29)	1,4 (55.5)	1,4 (33.1)	1,2 (0.007)	1,2 (0.007)	1,2 (14)	1,4 (52)
1,4 (5.52)	1,2 (48.9)	1,2 (9.4)	1,4 (0.015)	1,4 (0.015)	1,4 (19)	1,2 (43)
2,4 (8.81)	2,4 (36.1)	2,4 (9.2)	2,4 (0.04)	2,4 (0.039)	2,4 (26)	2,4 (36)
2,3 (10.92)	1,3 (18.7)	1,3 (6.9)	2,3 (0.066)	2,3 (0.066)	2,3 (29)	1,3 (27)
1,3 (14.21)	3,4 (14.6)	3,4 (6.6)	1,3 (0.104)	1,3 (0.099)	1,3 (46)	2,3 (25)
3,4 (19.73)	2,3 (11.8)	2,3 (5.9)	3,4 (0.208)	3,4 (0.188)	3,4 (78)	3,4 (11)

TABLE 6.7: Ranking of all pairs for the Legal document according to instance-wise metrics.

6.4.8 Reordering

This metric is based on the amount of reordering of the permutations needed in order to transform the sentence of one post-editor to the sentence of another.

We use the **Kendall's Tau distance** which, for two permutations π, σ (which correspond to the sentences of two post-editors) is calculated as

$$d_\tau(\pi, \sigma) = 1 - \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{n} z_{ij}}{Z}$$

where $z_{ij} = 1$ if $\pi(i) < \pi(j)$ and $\sigma(i) > \sigma(j)$ and 0 otherwise, and

$$Z = \frac{(n^2 - n)}{2}$$

Quoting Birch et al. (2010) from *"Metrics of MT Evaluation: Evaluating Reordering"*, Kendall's Tau:

> "... reflects the sum of all pairwise differences in order between the two permutations. The Kendall's tau metric is sensitive to relative orderings between all pairs of words and therefore to the distance that words are reordered. It can be interpreted as the probability that pairs of items in two different permutations are in the same order as opposed to being in different orders."

The permutations are computed using the alignments created by the *tercpp* library. The unaligned words (in the cases of Insertion and Deletion) are aligned to the previously aligned words, following Birch's guidelines. The result of the metric is the average reordering over all the instances. In order to make the results more distinguishable, we only average over the sentences that indeed need to be reordered, excluding from the calculation the sentences where $d_\tau = 1$, meaning that no reordering is needed.

The resulting ranking is shown in table 6.8:

HTER diff	**n-grams**	**Norm.Av. Overlap**	**Distr. Diff. (Hellinger)**	**#times Different**	**#times Similar**	**Reordering (Kendall)**
1,2 (3.29)	1,4 (55.5)	1,4 (33.1)	1,2 (0.007)	1,2 (14)	1,4 (52)	1,4 (0.979)
1,4 (5.52)	1,2 (48.9)	1,2 (9.4)	1,4 (0.015)	1,4 (19)	1,2 (43)	2,4 (0.978)
2,4 (8.81)	2,4 (36.1)	2,4 (9.2)	2,4 (0.039)	2,4 (26)	2,4 (36)	1,2 (0.974)
2,3 (10.92)	1,3 (18.7)	1,3 (6.9)	2,3 (0.066)	2,3 (29)	1,3 (27)	3,4 (0.968)
1,3 (14.21)	3,4 (14.6)	3,4 (6.6)	1,3 (0.099)	1,3 (46)	2,3 (25)	2,3 (0.967)
3,4 (19.73)	2,3 (11.8)	2,3 (5.9)	3,4 (0.188)	3,4 (78)	3,4 (11)	1,3 (0.967)

TABLE 6.8: Ranking of all pairs for the Legal document according to permutation metrics.

6.4.9 Ranking Results

The rankings for most metrics (some are excluded for space reasons) for the post-editors of the Legal document can be found in table 6.8.

First thing to note for the L document is that all metrics agree as to which post-editors are more similar. The pairs $(1,4),(1,2)$ and $(2,4)$ are always on the top 3 rows of the table. It is not clear, though, which one of these pairs is the most similar. However, one could claim that it is possible to identify the most different pair of post-editors to be pair $(3,4)$.

The results for the IT document (shown in table 6.9) are more ambiguous. The differences of the results are much smaller, compared to the L document, making it more difficult to distinguish between the pairs. The most probable *similar* pair could be $(3,4)$, despite the fact that it ranks last in the reordering metric.

HTER diff	**n-grams**	**Norm.Av. Overlap**	**Distr. Diff. (Hellinger)**	**#times Different**	**#times Similar**	**Reordering (Kendall)**
3,4 (1.12)	3,4 (65.5)	2,4 (40.1)	3,4 (0.001)	3,4 (63)	3,4 (102)	1,2 (0.947)
1,3 (1.60)	1,4 (48.8)	3,4 (23.4)	1,3 (0.001)	1,3 (75)	1,4 (71)	1,3 (0.947)
1,4 (2.72)	1,3 (43.3)	1,3 (22.4)	1,4 (0.005)	1,4 (82)	1,3 (68)	2,3 (0.942)
1,2 (8.45)	2,4 (39.3)	2,3 (20.6)	1,2 (0.021)	2,4 (83)	2,4 (63)	2,4 (0.942)
2,3 (10.05)	2,3 (28.8)	1,4 (13.9)	2,3 (0.031)	2,3 (87)	1,2 (54)	1,4 (0.941)
2,4 (11.17)	1,2 (23.5)	1,2 (12.8)	2,4 (0.041)	1,2 (89)	2,3 (51)	3,4 (0.939)

TABLE 6.9: Ranking of all pairs for the IT document according to various metrics.

Finding the most *different* pair would be an even more difficult task. In the *#times Different* metric, where we count the number of instances for which each pair has the most different labels, all pairs get comparable results, around the $\frac{1}{3}$ or $\frac{1}{4}$ of all instances.

(Note: In comparison, for the L document, the most similar pair has the highest HTER difference for around $\frac{1}{8}$ of the instances, whereas the most different pair has the highest HTER difference in the $\frac{1}{2}$ of all instances.)

6.4.9.1 Ranking Results Across Domains

When changing domain, some of the metrics cannot be applied (*reordering, instance-wise HTER difference ranking*, probably even *overlap* and *n-grams*[2]), since the post-editors work on completely different sentences.

Thus we have to use measures of distributions' similarity.

The results for the n-grams using all words (computed over 2-,3- and 4-grams, since there are no 5-grams in common in most cases) represent in general a small fraction of the vocabulary and are not really informative. In this case, it is probably more informative to use the n-grams of non-content words and punctuation; this result is what is shown in tables 6.10 and 6.11.

In addition, the average overlap, as expected, is in most cases less than one word and does not provide any information at all.

6.4.10 Conclusion

Based on the previous results, we can make the following observations:

- Most metrics seem to roughly agree as to which post-editor pairs are more similar or more different.

[2]The metrics of *overlap* and *n-grams* could indeed be computed, but the results in our case indicate very small overlap of words or n-grams between sentences of the two datasets. For example, the average overlap is around 1.2 words for Legal-IT sentences.

HTER diff	**n-grams**	**Avg. Overlap**	**Distr. Diff. (Hellinger)**	**Distr. Diff. (Bhatta)**
3,1 (3.93)	1,1 (58.0)	3,2 (0.90)	2,4 (0.006)	2,4 (0.006)
2,4 (4.27)	3,4 (51.5)	2,1 (0.87)	2,3 (0.011)	2,3 (0.011)
3,2 (4.52)	2,4 (46.2)	4,2 (0.86)	2,1 (0.018)	2,1 (0.018)
...	...	...	...	...
4,1 (15.80)	3,1 (24.4)	1,4 (0.49)	4,1 (0.105)	4,1 (0.110)
1,2 (18.73)	4,1 (24.1)	1,3 (0.49)	1,2 (0.126)	1,2 (0.134)
4,2 (24.25)	4,2 (21.8)	1,1 (0.47)	4,2 (0.211)	4,2 (0.236)

TABLE 6.10: Ranking of most similar and most different pairs for L-IT documents according to various metrics.

HTER diff	**n-grams**	**Avg. Overlap**	**Distr. Diff. (Hellinger)**	**Distr. Diff. (Bhatta)**
1,3 (3.93)	4,1 (91.8)	3,3 (0.91)	4,2 (0.006)	4,2 (0.006)
4,2 (4.27)	2,1 (86.1)	2,2 (0.88)	3,2 (0.011)	3,2 (0.011)
2,3 (4.52)	3,1 (66.2)	2,4 (0.88)	1,2 (0.018)	1,2 (0.018)
...	...	...	...	...
1,4 (15.80)	2,3 (22.8)	3,2 (0.49)	1,4 (0.105)	1,4 (0.110)
2,1 (18.73)	2,4 (22.0)	3,1 (0.49)	2,1 (0.126)	2,1 (0.134)
2,4 (24.25)	4,4 (19.1)	4,1 (0.50)	2,4 (0.211)	2,4 (0.236)

TABLE 6.11: Ranking of most similar and most different pairs for IT-L documents according to various metrics.

metric	**Correlation with other metrics**
HTER diff	0.699
n-grams	0.609
Avg. Overlap	0.585
Hellinger Distance	0.589
#times similar	0.284
#times different	0.632
reordering	0.389

TABLE 6.12: Correlation of post-editor similarity metrics with the other metrics

- Average HTER difference could be an adequate metric of post-editing behaviour differences, since it is in line with more of the other suggested metrics.

In order to justify the second claim, we compute the correlation of the scores that the metrics produce, for all possible pairs of metrics. Then for each metric, we average its correlation with the rest of the metrics and the results are shown in table 6.12.

Of course, one would require more rich models, that could use more rich features, especially linguistic ones, to accurately define a post-editor's behaviour. For example, a tendency of a post-editor to nominalise, compared to the preference of another post-editor to use more verbs, could only be captured by using, eg. a Part-of-Speech tagger.

However, a metric like HTER, which incorporates different types of edits (insertion of words, deletion, substitution, reordering) is adequate to determine pairs of translators with similar or different behaviour.

6.5 Changing Post-Editor in the same Domain

This setting explores the performance of the *online* and *batch* algorithms in the QE task, when the *training* and *test* sets are the result of the work of different post-editors on the same document.

In this setting, the *training* and *test* sets come from the same domain and the features are extracted using resources according to the respective domain.

According to the way they are created, the datasets allow us to evaluate the reactivity of different models when *training* and *test* data from the same domain are post-edited by different users, who show either similar (thus resulting in some degree of similarity in the distribution of the labels) or different behaviour (thus resulting in a smaller degree of similarity in the distribution of the labels).

6.5.1 Experimental Setup

For each document D (L or IT), the creation of the *training* and *test* sets is done as follows. D, for which post editions by four translators are available, is divided in two parts of equal size (80 instances for L and 140 for IT). This results in one training and one test set for each post editor. Note that the instances are selected in such a way, that all *training* sets (and, respectively, all *test* sets) consist of the same instances, across post-editors. The labels, that result from each post-editor's work are, of course, different.

For each learning algorithm (ALG) and all the combinations of users (i,j), it is now possible to evaluate ALG^i_j. Based on the predictions of ALG^i_j we can calculate the overall MAE of each model and the pointwise error (difference between the predictions and the true HTER labels for each instance of the test set). Parameters i and j can take the values $[1, 2, 3, 4]$, according to the corresponding translators.

Since, as shown in §5.7 there is no significant difference between the two kernels (RBF and Linear), for both *OSVR* and *OGP* we only use the RBF kernel in this set of experiments.

6.5.2 Experiment 3: Post-Editors within the same domain

The results for these experiments will be presented independently for each translation domain (IT and Legal).

6.5.2.1 IT domain

In Table 6.13 we present the results for training and testing on all combinations of post-editors, for the IT document. The post-editors in the rows correspond to *training*, whereas the post-editors in the columns correspond to *test*. In this setting:

1. Both *batch* and *online* methods perform well, depending on the difference of the post-editors. When the post-editors are similar, as in cases (1,1) or (1,3), the best results are achieved by SVR (*batch*), however the difference with *online* methods is not statistically significant. In the other hand, when the post-editors are more different, as in cases (3,1) or (3,2), *online* methods yield significantly better results.
2. *OSVR* and *OGP* consistently outperform *PA* (with the exception of only one case), however it is not clear which of the two performs better overall.
3. The *empty* configuration results are always worse than the *adaptive* and *batch*.

6.5.2.2 Legal domain

In Table 6.14 we present the results for training and testing on all the combinations of post-editors, for the Legal document.

1. Again, both *batch* and *online* methods perform well, depending on the difference of the post-editors. In almost all cases, the best results are achieved by *online* methods.
2. *OSVR* and *OGP* consistently outperform *PA*.
3. The *empty* configuration results are always worse than the *adaptive* and *batch*.

6.5.3 Results Discussion

Following the results presented in §6.5.2 for the challenging scenario where training and test data for each domain are obtained from two different post-editors, we can draw certain conclusions, regarding the performance of *online* and *batch* methods.

Batch (SVR)				
	1	2	3	4
1	**16.16**†	18.14	**14.58**	14.95
2	17.02	**15.97**†	16.08	16.73
3	16.62	20.00	14.63	14.98
4	16.65	20.09	14.64	14.98
Adaptive				
OSVR				
	1	2	3	4
1	16.49	17.14	14.74	15.17
2	**16.35**	**16.03**	**15.25**	15.60
3	16.64	17.28	14.76	15.03
4	16.57	18.06	**14.52**	15.12
empty	16.76	16.56	15.58	15.56
PA				
	1	2	3	4
1	18.17	22.99	17.17	15.97
2	18.15	23.01	17.15	**15.53**
3	18.06	22.95	16.95	15.37
4	18.01	22.87	16.51	15.86
empty	19.23	23.46	18.40	16.61
OGP				
	1	2	3	4
1	**16.16**	**16.81**	14.80	**14.79**
2	16.68	16.17	15.72	15.76
3	**15.87**	**17.17**	**14.52**	**14.49**
4	**16.13**	**17.25**	14.65	**14.69**
empty	18.28	18.34	16.73	16.33

TABLE 6.13: MAE of *batch*, *adaptive* and *empty* models on IT document for all pairs of post-editors. The best performing algorithm for each pair is denoted in bold.

As shown in tables 6.13 and 6.14, global MAE scores for the online algorithms indicate their good adaptation capabilities in all cases.

From the previously shown results, it is evident that all *online* methods perform better in the Legal domain, than in the IT domain. This also holds for the *batch* method.

In general, the performance of the methods depends heavily on the difference of the datasets. Selecting the best performing *adaptive* methods for each translator pair (as shown in tables 6.15 and 6.16), we calculate the correlation[3] between the performance (in terms of MAE) and the absolute difference of the average HTER label of the two datasets used for *training* and *test*. The same correlation is computed for *SVR*, the *batch* method. The results are shown in table 6.17.

It is obvious that *batch* methods are more affected from the difference of the datasets. The *online* methods are less affected by this difference, indeed confirming their capability

[3]With the Pearson correlation coefficient(Asuero et al., 2006).

Batch (SVR)				
	1	2	3	4
1	13.28	**14.68**†	16.56	13.93
2	13.48	14.85	15.77	14.55
3	17.02	16.57	10.58	21.09
4	13.63	16.56	23.08	12.01
Adaptive				
OSVR				
	1	2	3	4
1	**13.26**	14.69	**13.12**	**12.69**
2	13.27	14.73	**12.57**	**13.2**
3	15.67	16.17	**10.39**	16.48
4	13.44	15.48	16.77	12.04
empty	13.82	15.81	11.74	12.83
PA				
	1	2	3	4
1	13.29	14.93	16.25	14.03
2	13.31	14.97	16.17	14.12
3	**13.36**	**15.04**	16.13	**14.3**
4	13.39	**15.16**	16.06	14.2
empty	16.24	18.24	18.6	16.64
OGP				
	1	2	3	4
1	12.96	14.89	13.85	12.78
2	**13.07**	**14.65**	13.11	13.44
3	14.71	15.43	10.44	16.7
4	**13.1**	15.25	**15.95**	**11.88**
empty	23.06	19.41	28.05	18.82

TABLE 6.14: MAE of *batch*, *adaptive* and *empty* models on the Legal document for all pairs of post-editors. The best performing algorithm for each pair is denoted in bold.

	1	2	3	4
1	16.16	16.81	14.74	14.79
2	16.35	16.03	15.24	15.53
3	15.87	17.17	14.52	14.49
4	16.13	17.25	14.52	14.69

TABLE 6.15: Performance in MAE for the best *Adaptive* method in the IT domain

	1	2	3	4
1	12.96	14.69	13.12	12.69
2	13.07	14.65	12.57	13.2
3	13.36	15.04	10.39	14.3
4	13.1	15.16	15.95	11.88

TABLE 6.16: Performance in MAE for the best *Adaptive* method in the Legal domain

Method	**Legal**	**IT**
batch	0.884	0.889
adaptive	0.422	0.729

TABLE 6.17: Correlation of the average HTER difference and the performance of *batch* and *online* methods

to adapt to differences in the datasets. It is worth noting that, due to the Legal domain having less training data, it is much easier for *online* methods to adapt to the test set, thus resulting in quite low correlation, as compared to the IT domain.

A closer look at the behaviour of the online algorithms in the two domains leads to other observations. As regards the L domain, *online* models almost always outperform SVR (*batch*) significantly. For the IT domain, the *batch* performs (not significantly) better in three cases (out of sixteen). In general, the performance of the *batch* algorithm is competent only in the cases where the difference of the dataset is minimal. In the other cases, the *online* algorithms perform significantly better.

The fact than *online* methods perform significantly better motivates this work and represents an important finding, especially when seen from the application-oriented perspective, considering the high costs of acquiring large and representative QE training data, which would enable *batch* methods to be competent.

The performance of the algorithms in the *empty* setting can be better investigated with the use of plots of the cumulative MAE for the two domains in figure 6.2. It is evident that the Legal test set poses additional difficulty especially for OGP in the cases of translators 1 and 3. This can be explained by the fact that its $\mathcal{BV}$ vector is not filled with informative instances. This obstacle is overcome in the *adaptive* setting, where the *training* set has provided some informative instances.

Considering the performance of each individual algorithm, now, PA does not always achieve a lower MAE than SVR, especially in the IT domain, showing a quite unstable behaviour. OGP and $OSVR$, though, constantly perform better (or not significantly worse) that SVR.

6.6 Changing User across Domains

This setting explores the performance of the *online* and *batch* algorithms in the QE task, when the *training* and *test* sets are the result of the work of different post-editors, not on the same document, like in §6.5, but on different documents of, in fact, different domains.

In this setting, the *training* and *test* sets come from different domains and the features used are extracted using the resources according to the domain that is used in training.

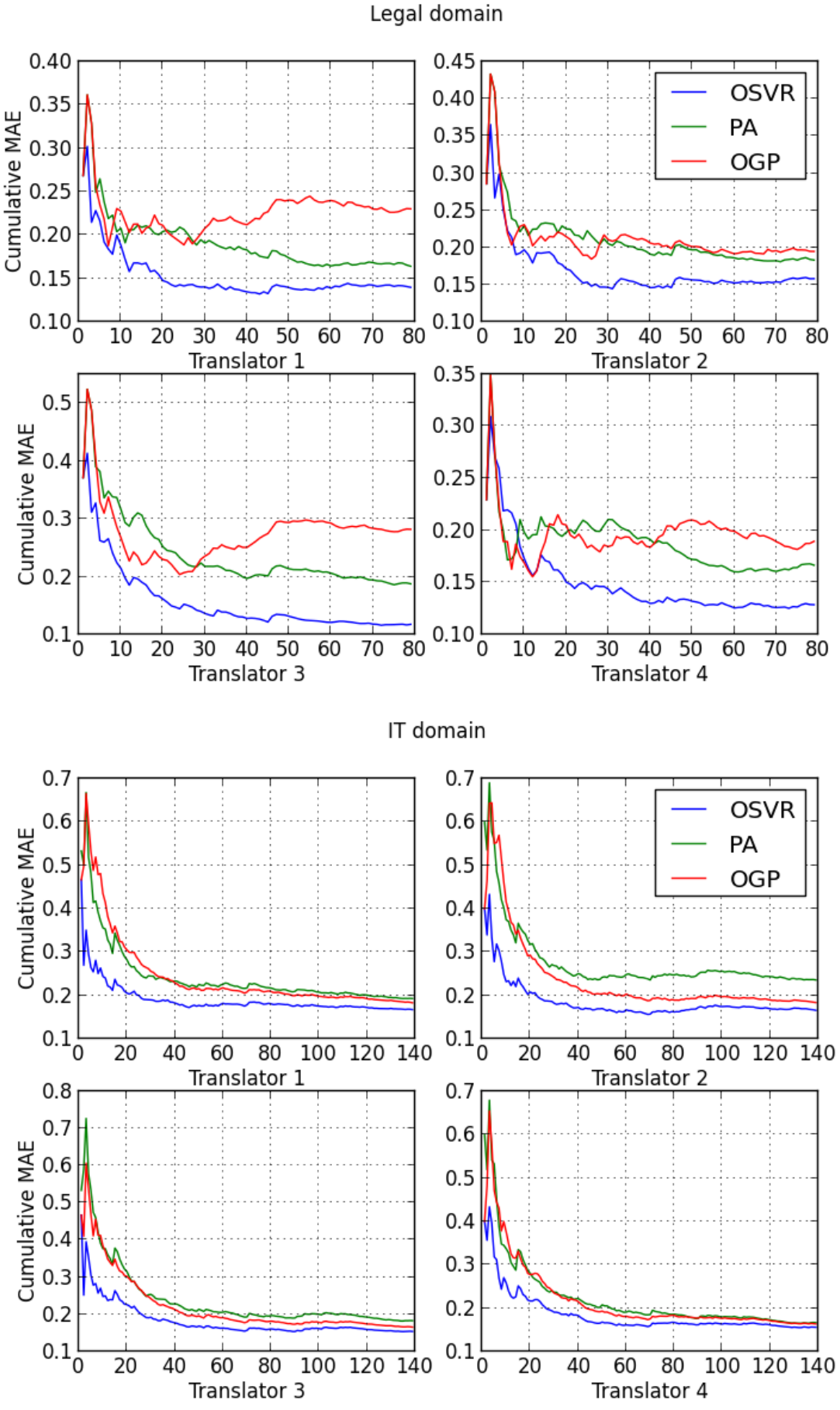

FIGURE 6.2: Cumulative MAE for the *empty* setting in the Legal and IT domain.

For example, if the *training* set comes from the IT domain, then for the features of both the *training* and *test* set, there are IT resources used, even if the *test* set comes from the Legal domain. Vice versa, if the *training* set is from the Legal document, then Legal resources are used for both the *training* and *test* sets, even if the *test* set comes from the IT document.

According to the way they are created, the datasets allow us to evaluate the reactivity of different models when *training* and *test* data from different domains are post-edited by different users, who show either similar behaviour (thus resulting in some degree of similarity in the distribution of the labels) or different behaviour (thus resulting in smaller degree of similarity in the distribution of the labels).

6.6.1 Experimental Setup

Following the steps of §6.5.1, for each document D (L or IT) and for each translator there are two sets created, by dividing the document in two parts of equal size (80 instances for L and 140 for IT). Note that, again, the instances are selected in such a way, so that all *training* sets (and, respectively, all *test* sets) consist of the same instances, across post-editors. The labels, though, that result from each post-editor's work are, of course, different.

In this setting we will use two of the post-editors in each domain. From the IT domain we will use the post-editors 2 and 4 (henceforth *High,IT* and *Low,IT*) which are the post-editors with the most different behaviour in this domain. In an equivalent way, we choose two translators from the Legal domain, post-editors 3 and 4 (henceforth *High,L* and *Low,L*).

The 8 resulting combinations of post-editors from different domains are ranked according to the average HTER difference of the datasets, as shown in table 6.18.

Experiment	*Training* Set	*Test* Set	HTER Diff.
4.1	Low,L	High,IT	24.5
4.2	High,IT	Low,L	24
4.3	Low,IT	Low,L	13.5
4.4	Low,L	Low,IT	12.7
4.5	Low,IT	High,L	8.3
4.6	High,L	High,IT	6.8
4.7	High,L	Low,IT	5
4.8	High,IT	High,L	2.2

TABLE 6.18: Set of experiments for Experiment 4.

It is important to mention that, for the *training* set, always the first split of the equivalent translator is used. In the same way, for the *test* set is always used the second split of the equivalent translator. This setting will also allow us to compare these experiments to the

situations where no domain change is happening (as in §6.5.2), enabling full comparison and understanding of the results.

Another factor that could probably affect the behaviour of all methods is the fact that, in this setting, the features of the *test* set are computed using resources from a different domain and not from its respective domain. For example, the features for the *test* set of experiment 5.1 are computed using Legal resources. In order to understand if this factor can actually play a crucial role, we have computed, for each experiment, the cosine difference of the features of the *training* set and the features of the *test* set. The results show minimal cosine difference, ranging from 0.006 to 0.043. These results mean that there is not an important difference between the features of the *training* and the *test* set, despite the fact that the resources used were only designed for the *training* set. Thus, the fact that we use different resources for the *test* set than the "appropriate" ones, cannot affect significantly the performance of the learning algorithms, or create the conditions for unstable behaviour.

The notation that is used in the next tables is $ALG_{i,C}^{j,D}$ where C, D refer to the domain of the *training* and *test* set. For this experiment, parameters i and j can take the values *Low* and *High*, referring to the two translators with the more different behaviour (the most conservative one, with the lowest average label - *Low* - and the most radical one, with the highest average label - *High* - respectively)

Similar to previous experiments, global MAE scores aim to compare the *adaptability* of each algorithm to heterogeneous test sets obtained from different users. We also introduce pointwise error plots, which, instead, aim at a more fine-grained analysis of the *sensitivity* of the *online* models to such differences.

Like in §5.4.1, the statistical significance of the results is calculated with approximate randomization.

6.6.2 Experiment 4: Post-Editors from different domains

In this section, the results of the experiments regarding a change of both post-editor and domain across *training* and *test* set, are presented. For the sake of brevity, only the best results from each mode are reported, also indicating the algorithm that yields these results. The detailed tables with the results of this experiments can be found in the appendix A.

The pointwise error plots show an 10-point-average moving MAE for all the experiments, for the *batch* and the best *adaptive* and *empty* systems. These are presented in figure 6.3‘.

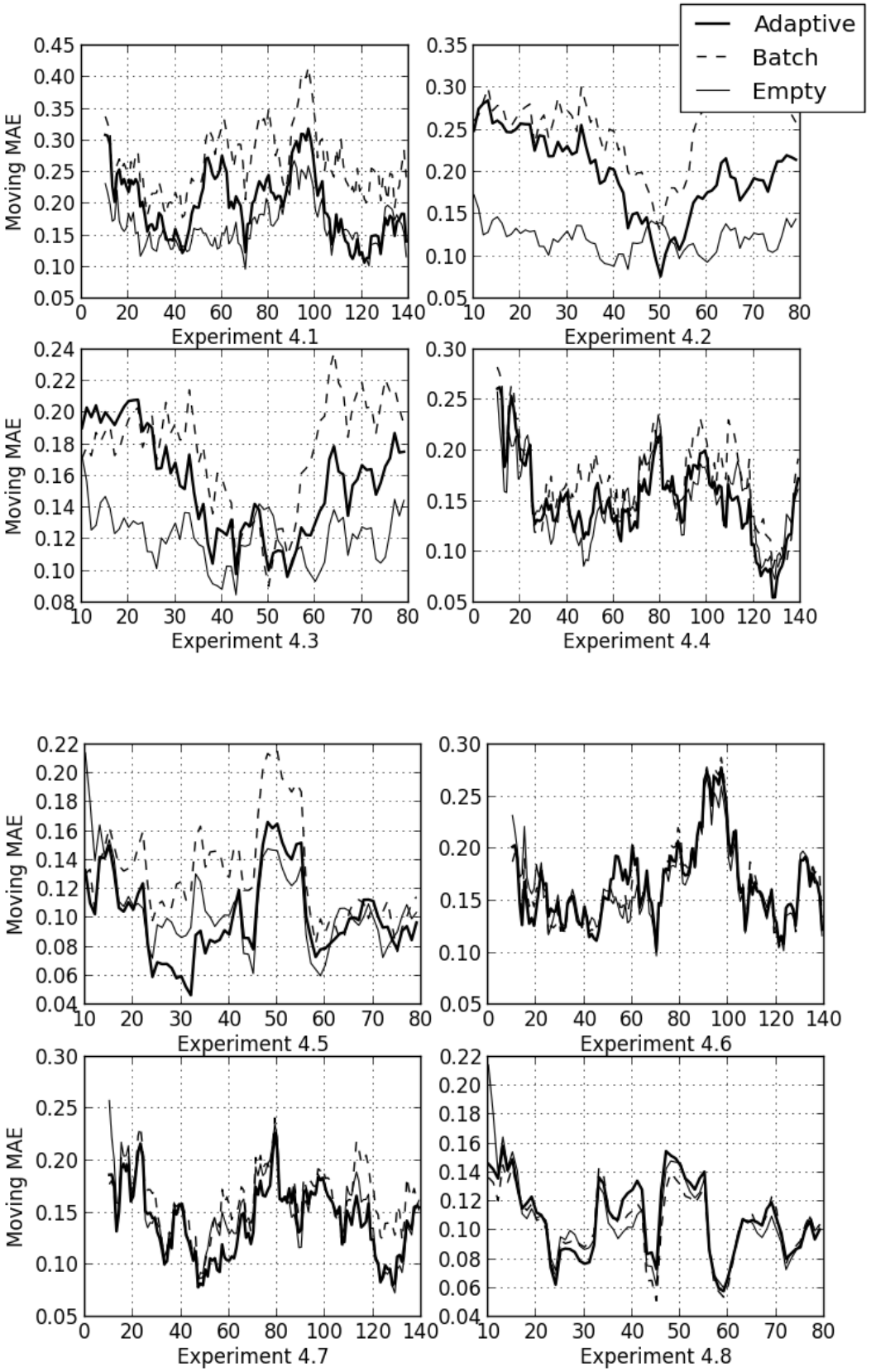

FIGURE 6.3: Sensitivity plots for Experiment 4.

Experiment		HTER Diff.	MAE Batch	MAE Adaptive	MAE Empty
4.1	$ALG^{High,IT}_{Low,L}$	24.5	27.00	19.77 ($OSVR$)	16.55 ($OSVR$)
4.2	$ALG^{Low,L}_{High,IT}$	24.0	25.37	19.96 (OGP)	12.46 ($OSVR$)
4.3	$ALG^{Low,L}_{Low,IT}$	13.5	17.54	15.73 ($OSVR$)	12.46 ($OSVR$)
4.4	$ALG^{Low,IT}_{Low,L}$	12.7	17.58	15.50 (OGP)	15.45 ($OSVR$)
4.5	$ALG^{High,L}_{Low,IT}$	8.3	13.00	10.51 (OGP)	11.28 ($OSVR$)
4.6	$ALG^{High,IT}_{High,L}$	6.8	16.89	16.38 ($OSVR$)	16.55 ($OSVR$)
4.7	$ALG^{Low,IT}_{High,L}$	5.0	16.15	14.40 (OGP)	15.45 ($OSVR$)
4.8	$ALG^{High,L}_{High,IT}$	2.2	10.84	10.64 ($OSVR$)	11.28 ($OSVR$)

TABLE 6.19: Results of Experiment 4.

In order to make the results even more visible, we fit a linear trend into the performance data[4] for all algorithms. The results are presented in figure 6.4. From the trend plots, we can draw some interesting conclusions:

- the performance of *batch* methods is in general stable, depending on the difference of the *training* and *test* set.
- in all cases, *adaptive* methods perform much better than the *batch* methods. They always show a downward trend and always achieve a better performance than *batch*. This improvement is more visible in the cases where the difference of the datasets is notable, resulting in very bad results for *batch* (experiments 4.1 through 4.4 - especially 4.1 and 4.2).
- *empty* methods perform quite well, especially in the cases were *training* might actually worsen the performance over the *test* set; that is, the cases were the difference of the datasets is too big.

In addition, we compute the correlation between the performance of the best algorithms in each sub-experiment and the difference of the average HTER label of the datasets. The results are presented in table 6.20 and show that both *batch* and *adaptive* are heavily affected by the difference of the datasets, but still *batch* show worse results. As expected, the *empty* models show no correlation to this difference.

[4]On the actual pointwise error, not on the moving average error.

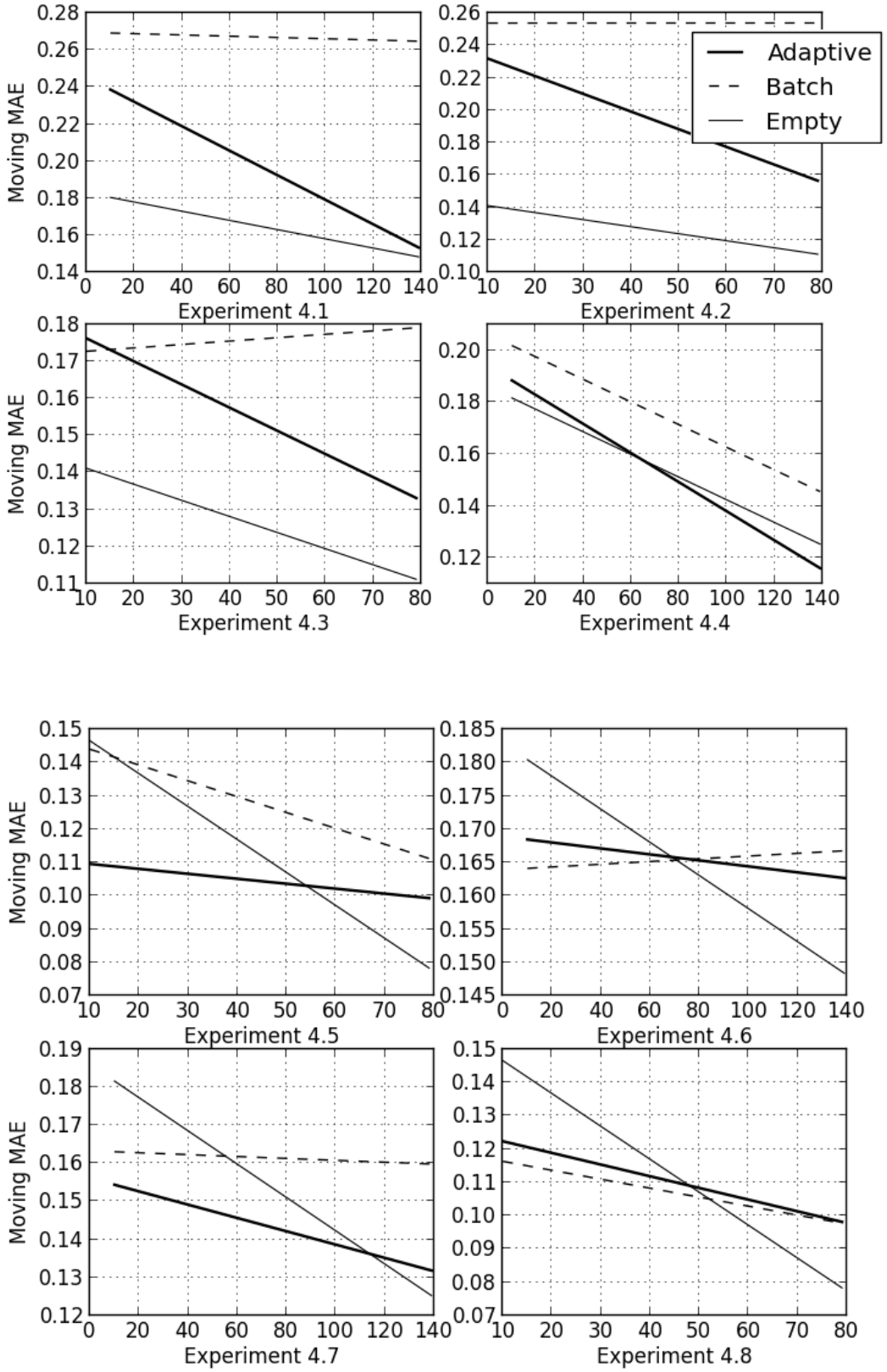

FIGURE 6.4: Trends of the sensitivity plots for Experiment 4.

Mode	Correlation
batch	0.945
adaptive	0.812
empty	0.190

TABLE 6.20: Correlation of performance and datasets difference for Experiment 4.

6.6.3 Results Discussion

The previous experiments allow us to draw some concrete conclusions.

First of all, when dealing with both changes of post-editor and changes of domain between training and testing, the *adaptive* models always have better performance than the *batch* models.

The degree of this improvement depends on the difference of the distributions of the HTER labels between the *training* and *test* sets. The higher this difference, the higher the improvement that is to be expected from using *adaptive* models. To put it in other words, the higher this difference, the worse the performance of the *batch* models.

In fact, when this difference is too big (having, for example, completely different distributions, like in experiment 4.1 or 4.2 in section §6.6.2), the best performance comes when using an *empty online* model. In such cases, the *training* data work against the *adaptive* model. Based on the results in table 6.19, one could identify an approximate pivot point around 13 HTER points of datasets' difference, beyond which *empty* models perform better than *adaptive* ones.

However, using an empty model every time a new translation job begins is not a plausible real scenario. To begin with, one wouldn't know that the eventual distribution of the *test* set would be different enough, in order to justify using an *empty* model. However, using an *adaptive* model would always make sense, since the *adaptive* models yield very good results in all the previous experiments and seem to respond quite fast to the new *test* instances.

Another definite conclusion we can draw, regards the behaviour of the *online* algorithms. For one thing, we can claim that *OSVR* and *OGP* show remarkable robustness and stability, performing reasonably well in all the experiments. *OSVR* though seems to perform slightly better (although, not significantly in all cases) also in the *empty* mode.

On the other hand, we can also support the claim that *PA* is quite unstable, capable for both performing as good as the other algorithms, but also for performing significantly worse, even when compared to *batch* methods (this is observed in experiments 4.3 and 4.6).

Chapter 7

Conclusion

7.1 Synopsis

This thesis attempted to introduce online methods for the task of Quality Estimation of MT output and to verify whether an adaptive quality estimation system would be beneficial for a CAT-tool.

Based on the previously presented experiments (§5,§6) we have shown that the performance of the systems depends on the difference of the datasets that are used for *training* and *testing*. In cases where no adaptation is needed, both *batch* and *online* methods achieve good results.

However, as the difference of the datasets increases, due to factors like a change of the post-editors or a change of the translation domain, adaptation is needed. In these cases, the *adaptive* methods perform significantly better than the *batch* ones.

In addition, we showed that the system (§4) that we created can be not only beneficial to a CAT-tool, but also easily integrated, as the resources in requires are already used by the CAT-tool. Furthermore, its response time (either for providing a prediction or updating the model) is quite small and would not affect negatively the productivity of the translator.

Finally, the suggested configuration for such a QE server (eg. as a default) would be the one using *Online Gaussian Processes*, because they are not only robust in terms of performance, but are also quite fast, compared to *Online SVR*. Should an *empty* mode be preferred, though, then *OSVR* prove to be more robust for building a model from scratch, as shown by experiment 4 ($6.6.2) Although *PA* is the fastest of the three methods that we examined, it would not be suggested because it shows unstable performance.

7.2 Further Work

Despite the substantial progress done so far in the field, there are aspects of research on Quality Estimation that still need to be investigated.

The use of *adaptive* methods for the QE task, in order to deal with certain challenges such as domain adaptation, has just been introduced.

The system that we developed for this thesis does not incorporate all the recent advances. The current approaches on the task focused more on feature engineering (e.g. alignment features (de Souza et al., 2013), combinatory categorial grammar features (Rubino et al., 2013), pseudo-references (Albrecht and Hwa, 2007)), model learning with a variety of classification and regression algorithms (e.g. SVR with partial least squares (Bicici, 2013), gaussian processes (Beck et al., 2013), M5P (Soricut et al., 2012)), and feature selection as a way to overcome sparsity and overfitting issues (Soricut et al., 2012)).

In our experiments we only used the 17 *baseline* features and did not incorporate any method for feature selection. Thus, one obvious step for further research is to investigate whether the addition of more features, along with a feature selection mechanism, would further boost the performance of the *online* algorithms. However, one should always keep in mind the *time* performance of the system. Certain features are time-consuming to extract, as are for example pseudo-references, which, in order to compute, require all the sentences to be translated. This is, therefore, another aspect that needs to be taken into account when one is working in an online fashion.

Such features could either be based on richer linguistic or statistical models, but they could also be more focused on the post-editor, the potential user of the CAT-tool. Such features could leverage a specific user's feedback, be stored as part of the QE model, and aid in tailoring the performance of the QE component to their individual quality standards. This step towards "personalisation" of the QE component would hopefully increase the post-editors productivity.

Moreover, as regards testing the performance of the QE component, developing additional corpora of *[src,trg,pe]* tuples for other language pairs and other translation domains is a crucial step for further exploration on this field. This could be achieved with the help of professional translation companies, either private or institutional. Increasing the number of available resources could only cause more interest on the field. Should such resources become available, we would like to test the performance of the *adaptive* methods also on these datasets.

Finally, another direction that could be investigated and that would be of interest, especially for the professional users of a CAT-tool, is the effort to determine the exact boundaries for which using either of a *batch, adaptive* or even *empty* model would be beneficial. Ideally, we would like to investigate the possibility of defining a metric that

could suggest, before the start of a translation job, which of the learning modes would produce the best results, according to the existing resources, the text to be translated, the domain, etc.

Appendix A

Detailed Results for Experiment 4

This section provides the results on the performance of all algorithms, in terms of MAE, in experiment 4. As in section §6.6.2, the experiments in tables A.1 and A.2 are ranked in descending order according to the difference of the average HTER label of the *training* and *test* datasets. The best performing algorithm is marked with *bold*.

Adaptive mode						
Train	**Test**	**Δ HTER**	***SVR***	***OSVR***	***OGP***	***PA***
Legal Low	IT High	24.5	27	**19.77**	19.92	21.73
IT High	Legal Low	24	25.37	20.58	**19.96**	24.25
IT Low	Legal Low	13.5	17.54	**15.72**	15.78	24.3
Legal Low	IT Low	12.7	17.58	15.70	**15.50**	16.8
IT Low	Legal High	8.3	13	12.16	**10.51**	15.98
Legal High	IT High	6.8	16.89	**16.38**	16.58	21.6
Legal High	IT Low	5	16.15	14.68	**14.40**	16.67
IT High	Legal High	2.2	**10.84**	10.64	11.19	16.17

Table A.1: Detailed results for Experiment 4 in *adaptive* mode.

Empty mode					
Train	**Test**	**Δ HTER**	***OSVR***	***OGP***	***PA***
Legal Low	IT High	24.5	**16.55**	18.34	23.46
IT High	Legal Low	24	**12.46**	18.82	16.65
IT Low	Legal Low	13.5	**12.46**	18.82	16.65
Legal Low	IT Low	12.7	**15.45**	16.33	16.60
IT Low	Legal High	8.3	**11.28**	28.05	18.60
Legal High	IT High	6.8	**16.55**	18.34	23.46
Legal High	IT Low	5	**15.45**	16.33	16.60
IT High	Legal High	2.2	**11.28**	28.05	18.60

Table A.2: Detailed results for Experiment 4 in *empty* mode.

Bibliography

Joshua Albrecht and Rebecca Hwa. Regression for sentence-level mt evaluation with pseudo references. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 296, 2007.

AG Asuero, A Sayago, and AG Gonzalez. The correlation coefficient: An overview. *Critical reviews in analytical chemistry*, 36(1):41–59, 2006.

Daniel Beck, Kashif Shah, Trevor Cohn, and Lucia Specia. SHEF-Lite: When less is more for translation quality estimation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 337–342, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2241`.

Nicola Bertoldi and Marcello Federico. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 182–189. Association for Computational Linguistics, 2009.

Nicola Bertoldi, Mauro Cettolo, and Federico Marcello. Cache-based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation. In *Proceedings of the XIV Machine Translation Summit*, pages 1147–1162, 2013.

Ergun Bicici. Feature decay algorithms for fast deployment of accurate statistical machine translation systems. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 78–84, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2206`.

Ergun Biçici, Declan Groves, and Josef van Genabith. Predicting sentence translation quality using extrinsic and language independent features. *Machine Translation*, 27 (3-4):171–192, 2013.

Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007.

Alexandra Birch, Miles Osborne, and Phil Blunsom. Metrics for mt evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26, 2010.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence Estimation for Machine Translation. Summer workshop final report, JHU/CLSP, 2003.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 315. Association for Computational Linguistics, 2004.

Ondrej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Eighth Workshop on Statistical Machine Translation*, WMT-2013, pages 1–44, Sofia, Bulgaria, 2013. URL `http://www.aclweb.org/anthology/W13-2201`.

Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F Zaidan. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64. Association for Computational Linguistics, 2011.

Nicolò Cesa-Bianchi, Gabriel Reverberi, and Sandor Szedmak. Online Learning Algorithms for Computer-Assisted Translation. Deliverable D4.2, SMART: Statistical Multilingual Analysis for Retrieval and Translation. 2008.

Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

Eugene Charniak, Kevin Knight, and Kenji Yamada. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46. Citeseer, 2003.

Jorge Civera and Alfons Juan. Domain adaptation in statistical machine translation with mixture modelling. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 177–180. Association for Computational Linguistics, 2007.

Trevor Cohn and Lucia Specia. Modelling Annotator Bias with Multi-task Gaussian Processes: An Application to Machine Translation Quality Estimation. 2013.

K Cramer, O Dekel, J KESHET, et al. Online passive-aggressive algorithm. *Journal of Machine learningResearch*, 7:551–585, 2007.

Josep M Crego and Nizar Habash. Using shallow syntax information to improve word alignment and reordering for smt. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 53–61. Association for Computational Linguistics, 2008.

Josep M Crego and José B Marino. Syntax-enhanced n-gram-based smt. *MT Summit XI*, pages 111–118, 2007.

Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

Hal Daumé III. Frustratingly easy domain adaptation. In *ACL*, volume 1785, page 1787, 2007.

Langlois David, Raybaud Sylvain, and Smaïli Kamel. Loria system for the wmt12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 114–119. Association for Computational Linguistics, 2012.

Robert Davies. Newmat c++ matrix library, 2000.

José Guilherme C. de Souza, Christian Buck, Marco Turchi, and Matteo Negri. FBK-UEdin participation to the WMT13 quality estimation shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 352–358, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2243`.

Michel Marie Deza and Elena Deza. *Encyclopedia of distances*. Springer, 2009.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc., 2002.

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

Marcello Federico, A Cattelan, and M Trombetti. Measuring user productivity in machine translation enhanced computer assisted translation. In *Tenth Biennial Conference of the Association for Machine Translation in the Americas*, 2012.

Michael Gamon, Anthony Aue, and Martine Smets. Sentence-level mt evaluation without reference translations: Beyond language modeling. In *Proceedings of EAMT*, pages 103–111, 2005.

Daniel H Grollman. *Teaching Old Dogs New Tricks: Incremental Multimap Regression for Interactive Robot Learning from Demonstration.* PhD thesis, Brown University, May 2010.

Christian Hardmeier. Improving machine translation quality prediction with syntactic tree kernels. In *Proceedings of the 15th conference of the European Association for Machine Translation (EAMT 2011)*, pages 233–240, 2011.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. Tree kernels for machine translation quality estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 109–113. Association for Computational Linguistics, 2012.

Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Prentice Hall New York, 2009.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. Bridging SMT and TM with Translation Recommendation. 2010.

Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in nlp. In *ACL*, volume 2007, page 22, 2007.

Philip Koehn. *Statistical Machine Translation*, volume 1. Cambridge University Press, 2010.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557769.1557821`.

Maarit Koponen. Comparing Human Perceptions of Post-editing Effort with Post-editing Operations. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 181–190. Association for Computational Linguistics, 2012.

Maarit Koponen, Wilker Aziz, Luciana Ramos, and Lucia Specia. Post-editing Time as a Measure of Cognitive Effort. In *Proceedings of the AMTA 2012 Workshop on Post-editing Technology and Practice (WPTP 2012)*, 2012.

Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics, 2007.

Dekang Lin. A path-based transfer model for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 625. Association for Computational Linguistics, 2004.

Nick Littlestone. Learning Quickly when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. In *Machine Learning*, pages 285–318, 1988.

Junshui Ma, James Theiler, and Simon Perkins. Accurate Online Support Vector Regression. *Neural Computation*, 15:2683–2703, 2003.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. Online Learning via Dynamic Reranking for Computer Assisted Translation. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part II*, CICLing'11, 2011.

Pascual Martínez-Gómez, Germán Sanchis-Trilles, and Francisco Casacuberta. Online adaptation strategies for statistical machine translation in post-editing scenarios. *Pattern Recogn.*, 45(9):3193–3203, September 2012. ISSN 0031-3203. doi: 10.1016/j.patcog.2012.01.011. URL `http://dx.doi.org/10.1016/j.patcog.2012.01.011`.

Prashant Mathur, Mauro Cettolo, and Marcello Federico. Online Learning Approaches in Computer Assisted Translation. In *Proceedings of the 8th Workshop on Statistical Machine Translation (WMT'13)*, Sofia, Bulgaria, 2013.

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

Tom Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the sixth international colloquium on cognitive science*, pages 2–11. Citeseer, 1999.

M. S. Nikulin. Chi-squared test for normality. In *Proceedings of the International Vilnius Conference on Probability Theory and Mathematical Statistics*, volume 2, pages 119–122, 1973.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics, 2002.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. Online learning for interactive statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 546–554, Stroudsburg, PA, USA, 2010. URL `http://dl.acm.org/citation.cfm?id=1857999.1858078`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Francesco Parrella. Online support vector regression. *Master's Thesis, Department of Information Science, University of Genoa, Italy*, 2007.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

Christopher Quirk. Training a sentence-level machine translation confidence measure. In *LREC*. Citeseer, 2004.

Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

Sylvain Raybaud, David Langlois, and Kamel Smaïli. "this sentence is wrong." detecting errors in machine-translated sentences. *Machine translation*, 25(1):1–34, 2011.

Raphael Rubino, Antonio Toral, S Cortés Vaıllo, Jun Xie, Xiaofeng Wu, Stephen Doherty, and Qun Liu. The cngl-dcu-prompsit translation systems for wmt13. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 211–216, 2013.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226, 2000.

D Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 979–988. ACM, 2010.

Anil Kumar Singh, Guillaume Wisniewski, and François Yvon. Limsi submission for the wmt'13 quality estimation task: an experiment with ngram posteriors. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 396–402, 2013.

Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006.

Radu Soricut, Nguyen Bach, and Ziyuan Wang. The SDL Language Weaver Systems in the WMT12 Quality Estimation Shared Task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation (WMT'12)*, pages 145–151, Montréal, Canada, 2012.

Lucia Specia. Exploiting objective annotations for measuring translation post-editing effort. In *Proceedings of the 15th Conference of the European Association for Machine Translation*, pages 73–80, 2011.

Lucia Specia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT'09)*, pages 28–35, Barcelona, Spain, 2009.

Lucia Specia, Dhwaj Raj, and Marco Turchi. Machine Translation Evaluation versus Quality Estimation. *Machine translation*, 24(1):39–50, 2010.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. QuEst - A Translation Quality Estimation Framework. In *51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, ACL-2013, pages 79–84, Sofia, Bulgaria, 2013. URL `http://www.aclweb.org/anthology/P13-4014`.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. The JRC-Acquis: a Multilingual Aligned Parallel Corpus with 20+ Languages. *CoRR*, abs/cs/0609058, 2006.

Jorg Tiedemann. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Marco Turchi, Matteo Negri, and Marcello Federico. Coping with the Subjectivity of Human Judgements in MT Quality Estimation. In *Proceedings of the 8th Workshop on Statistical Machine Translation (WMT'13)*, Sofia, Bulgaria, 2013.

Vladimir Vapnik, Steven E Golowich, and Alex Smola. Support vector method for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems*, pages 281–287, 1997.

Warren Weaver. Translation. *Machine translation of languages*, 14:15–23, 1955.

Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. 1996.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics, 2001.

Alexander Yeh. More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, pages 947–953, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.